WL-TR-93-1004

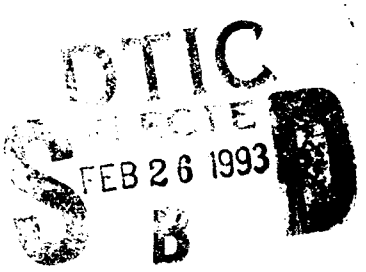
ACEC & AES MERGER WORKSHOP REPORT

RAYMOND SZYMANSKI

WRIGHT LABORATORY
AVOINICS DIRECTORATE
WL/AAAF-2
WRIGHT-PATTERSON AFB OH 45433-7409

MAR 1992

FINAL REPORT FOR 01/22/92-01/24/92



AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT PATTERSON AFB OH 45433 - 7409

NOTICE

WHEN GOVERNMENT DRAWINGS, SPECIFICATIONS, OR OTHER DATA ARE USED FOR ANY PURPOSE OTHER THAN IN CONNECTION WITH A DEFINITELY GOVERNMENT-RELATED PROCUREMENT, THE UNITED STATES GOVERNMENT INCURS NO RESPONSIBILITY OR ANY OBLIGATION WHATSOEVER. THE FACT THAT THE GOVERNMENT MAY HAVE FORMULATED OR IN ANY WAY SUPPLIED THE SAID DRAWINGS, SPECIFICATIONS, OR OTHER DATA, IS NOT TO BE REGARDED BY IMPLICATION, OR OTHERWISE IN any MANNER CONSTRUED, AS LICENSING THE HOLDER, OR ANY OTHER PERSON OR CORPORATION; OR AS CONVEYING ANY RIGHTS OR PERMISSION TO MANUFACTURE, USE, OR SELL ANY PATENTED INVENTION THAT MAY IN ANY WAY BE RELATED THERETO.

THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION.

Raymond Szymanski

RAYMOND SZYMANSKI
Program Manager

TIMOTHY G. KEARNS, Maj, USAF
Chief
Readiness Technology Group

Charles H Krueger

CHARLES H. KRUEGER, Chief
System Avionics Division
Avionics Directorate

IF YOUR ADDRESS HAS CHANGED, IF YOU WISH TO BE REMOVED FROM OUR MAILING LIST, OR IF THE ADDRESSEE IS NO LONGER EMPLOYED BY YOUR ORGANIZATION PLEASE NOTIFY WL/AAAF, WRIGHT-PATTERSON AFB, OH 45433-7409 TO HELP MAINTAIN A CURRENT MAILING LIST.

COPIES OF THIS REPORT SHOULD NOT BE RETURNED UNLESS RETURN IS REQUIRED BY SECURITY CONSIDERATIONS, CONTRACTUAL OBLIGATIONS, OR NOTICE ON A SPECIFIC DOCUMENT.

AGENCY USE ONLY (Leave blank)	REPORT DATE MAR 1992	REPORT TYPE AND DATES COVERED FINAL 01/22/92--01/24/92

TITLE AND SUBTITLE: ACEC & AES MERGER WORKSHOP REPORT

FUNDING NUMBERS
C
PE 63226
PR AJPO
TA 28
WU 53

AUTHOR(S): RAYMOND SZYMANSKI

PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
WRIGHT LABORATORY
AVOINICS DIRECTORATE
WL/AAAF-2
WRIGHT-PATTERSON AFB OH 45433

PERFORMING ORGANIZATION REPORT NUMBER

SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE SYSTEMS COMMAND
WL/AAAF, Attn: SZYMANSKI 513-2553947

SPONSORING MONITORING AGENCY REPORT NUMBER
WL-TR-93-1004

SUPPLEMENTARY NOTES
ADA JOINT PROGRAM OFFICE
1211 S. FERN ST
ARLINGTON VA 22202

12a. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

12b. DISTRIBUTION CODE

ABSTRACT (Maximum 200 words)

This report summarizes the activities and results of the ACEC & AES Merger Workshop which was conducted 22-24 January 1992. The purpose of the workshop was to discuss the issues pertinent to the merging of two distinct Ada compiler evaluation test suites. This report contains the recommendations provided by the participants and the supporting rationale for each recommendation. Also included is the presentation material used during the workshop and a list of the participants.

SUBJECT TERMS
ADA, ADA COMPILER EVALUATION CAPABILITY, ACEC, ADA EVALUATION SYSTEMS, AES, ADA PROGRAMMING SUPPORT ENVIRONMENT, APSE, EVALUATION, VALIDATION

NUMBER OF PAGES 110

PRICE CODE

SECURITY CLASSIFICATION OF REPORT	SECURITY CLASSIFICATION OF THIS PAGE	SECURITY CLASSIFICATION OF ABSTRACT	LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

ACKNOWLEDGEMENTS

The author is indebted to all the Workshop participants for their accomplishments during this event which has produced a significant milestone in the Ada program. Because of the expertise and dedication of these individuals, the Workshop results have contributed substantially to the overall goals of the ACEC & AES Merger activity. Additionally, I extend my gratitude to George Robertson, Lloyd Stiles and their support staff at NCCOSC, San Diego, for hosting the Workshop and providing critical administrative and technical support. Special thanks also goes to Barbara Rhoads and Trudy Grube who patiently listened, recorded and transcribed the proceedings for use in developing this report.

PREFACE

In June 1991, the Department of Defense of the United States of America (US DoD) and the United Kingdom Ministry of Defence (UK MOD) agreed to merge their respective Ada compiler evaluation test suites and produce a single, internationally available suite.

The development of the US suite, the Ada Compiler Evaluation Capability (ACEC), is managed by Mr. Raymond Szymanski of Wright Laboratory and was accomplished by Boeing Defense and Space Group Product Support Division, Wichita, Kansas. This effort is sponsored by the Ada Joint Program Office (AJPO).

The development of the UK suite, the Ada Evaluation System (AES) , was accomplished by the British Standards Institute and Software Sciences Ltd. This effort was sponsored by the United Kingdom Ministry of Defence.

The agreement between the two governments established the AJPO as the office responsible for carrying out the merger. As a result, the ACEC contractor developed an initial approach to merging the two suites after completing a substantial study of the AES. Their observations and analysis were used as a starting point for Merger Workshop discussions.

The Workshop co-chairs were Mr. Raymond Szymanski of Wright Laboratory, Evaluation & Validation Program Manager, and Mr. Dan Roy, of the Software Engineering Institute, Real-time Embedded Systems Testbed (REST) Project Leader.

The accomplishments of the Workshop are an important contribution to defining the merged product. Improvements in portability, usability, and completeness are expected as a result of the recommendations made and issues addressed.

Table of Contents

1.0 EXECUTIVE SUMMARY

1.1 Background

Dr John Solomond, AJPO Director, decided that the Ada community would be best served by a single Ada compiler evaluation suite. This test suite would embody the best capabilities of two government developed test suites, the ACEC of the US DoD, and the AES of the UK MoD.

In June 1991, the governments agreed to a suite merger with the US DoD's AJPO responsible for the activity. The AJPO, in turn, tasked Mr. Raymond Szymanski, the E&V Project manager, with management responsibility for the merged product. Mr. Szymanski is also responsible for the Planned Product Improvement cycle on the existing ACEC.

As specified in the merger agreement, the merged suite will have unlimited distribution to the Ada community both in the US and internationally.

1.2 Workshop Initiation

The success of the ACEC, as measured by over 200 users and the benefits derived from its use, is attributable to several management practices and technical factors employed during its development. These practices and factors include: product peer review during development, user evaluation between releases, selection of technically competent developers and reviewers, and team membership consistency. Additionally, participants from each user sector, government, industry and academia, provided the necessary multiple perspectives to insure all user needs were considered.

Following this successful formula, workshop invitees included the developers of the ACEC & the AES, a variety of users of both suites, compiler vendors, and independent evaluators of the ACEC & AES. This mix ensured that both test suites, as well as many types of future merged-suite users, were well represented.

1.3 Workshop Philosophy

The workshop was organized in a fashion to permit the participants to re-orient themselves to the details of both test suites and hear independent assessments of each. To allow this, the developers of both the ACEC and the AES were invited to present details of their suite's latest version, and plans for future versions. Individuals who have used either one or both suites were invited to present their findings of each suite's strengths and weaknesses. Additionally, the merger project office was invited to

present a proposed approach to merging the two suites. These presentations would then be followed by nearly two days of discussion on the merger subject.

1.4 Workshop Accomplishments

The debates and discussions that ensued during the workshop were vigorous and informing. As a result, the workshop succeeded in providing numerous positive recommendations relevant to merging the ACEC and AES. Some of those recommendations are listed below in a non priority order. Details of these recommendations, related issues and relevant discussion are contained in the main document sections and the appendices.

- Recommended "portability", "usability", and "completeness" as primary requirements for the merged product.

- Recommended the merged product to be an ACEC adaptation of AES technology and functionality.

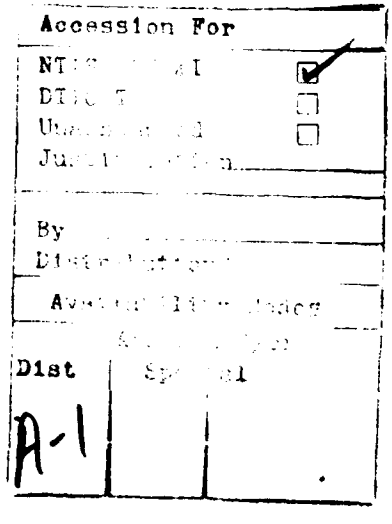
- Recommended a level of technical review be performed on AES elements prior to inclusion in the merged suite.

- Recommended the E&V Reference System as the future home for all AES non-compiler-related assessors.

- Recommended a priority for the integration of AES compiler-related assessors

- Recommended user interface improvements

- Recommended analysis reporting improvements



2.0 ACEC & AES MERGER WORKSHOP PROCEEDINGS

This section provides a summary of the workshop presentations, merger issues, discussions and merger recommendations.

2.1 Opening Remarks

The ACEC/AES Merger Workshop opened with a welcome and an introduction to the area by Mr. George Robertson, of Fleet Combat Directional Systems Support Activity (FCDSSA), San Diego. Mr. Robertson detailed the Navy's commitment to the Ada Language and the challenges they faced during the upcoming transition years. In closing, Mr. Robertson described FCDSSA'a usage of the ACEC test suite during evaluation of the Navy's Ada Language System / Navy (ALS/N).

Mr. Raymond Szymanski, Merger Workshop co-chair, welcomed everyone on behalf of Dr. John Solomond, AJPO Director and Mr. Dan Roy, fellow co-chairman. He stated the main objective of the workshop was to review and refine an approach to merging the ACEC and the AES. He reminded the participants that they were chosen on the bases of their technical expertise, their ability to work cooperatively as part of the merger team, and for the different professional perspectives they could provide. He reviewed the proposed agenda for the meeting and briefly discussed the various presentations that would be forthcoming.

Mr. Dan Roy commented on the need for the workshop attendees to consider cultural differences in the development of the two suites in addition to considering the views of users, vendors, and governments. He stated that his philosophy for the workshop is that the process of evaluation is more important than the specifics of the technology.

2.2 Summary of Presentations

The workshop was organized in a fashion to permit the participants to re-orient themselves to the details of both test suites and hear independent assessments of each. To allow this, the developers of both the ACEC and the AES were invited to present details of their suite's latest version, and plans for future versions. Individuals who have used either one or both suites were invited to present their findings of each suite's strengths and weaknesses. Additionally, the merger project office was invited to present a proposed approach to merging the two suites. These presentations are summarized below. The presentation vu-foils can be found in the appendices.

Note: Unfortunately, the AES developers were unable to attend the Workshop. However, AES information was presented by other attendees.

2.2.1 ACEC Version 3.0 Product - Boeing

Sam Ashby, Kermit Terrell, and Barbara Decker-Lindsey of the Boeing Defense and Space Group Product Support Division provided a status report on ACEC Version 3.0 development and a comprehensive presentation on ACEC Version 3.0 capabilities.

Mr. Ashby reported that ACEC Version 3.0 testing has been completed on five target systems, including the DEC self-hosted Ada, Telesoft VAX self-hosted Ada, VAX hosted compiler targeted to a 1750A processor (TLD), Meridian DEC Station self-hosted (UNIX), and Verdix self-hosted Silicon Graphics. All problems identified during testing were resolved and delivery was made to the customer on 18 December 1991. Mr. Ashby concluded by stating that the product is currently undergoing final customer review.

Mr. Terrell's presentation detailed the improvements made to the ACEC in Version 3.0 which included test suite reorganization, additional performance tests, new and expanded assessors, a new pre-test capability, and an enhanced user interface.

Ms. Decker-Lindsey's presentation detailed the capabilities of the ACEC Version 3.0 analysis tools and explained where improvements were made in analysis tool capabilities and user interfaces. These improvements include a user menuing system, an editable results data base, a data extraction tool, and a reduction in the number of steps required to perform the analysis.

2.2.2 AES Version 2.0 Assessment - Boeing

Mr. Tom Leavitt presented a review of the AES Version 2.0, based on his experience with running the AES, focusing specifically on the test harness, specific performance tests, assessors, and analysis and reporting capabilities. Mr. Leavitt's activities for this review included reading the documentation, examining the source code for the performance tests, executing the performance test groups, and running other selected AES elements.

2.2.3 AES Version 2.0 Assessment - SEI

Mr. Neal Altman presented a review of the AES Version 2.0, based on his experience with running the AES, focusing on the executable benchmark tests and the test harness as primary concerns, with the checklists and documentation as secondary concerns. He also discussed AES organizational issues and features.

2.2.4 ACEC Version 3.0 Assessment - SEI

Mr. Patrick Donohoe presented his experience with ACEC Version 3.0. He discussed the suite's documentation and the pre-test setup steps. He also discussed the performance tests that he had run to date.

2.2.5 E&V Reference System Version 3.1 Demonstration - TASC

Dr. Bard Crawford gave a demonstration of the Evaluation and Validation (E&V) Reference System Version 3.1 which is implemented with a hypertext capability. During this presentation he discussed the purpose of the E&V Reference System and demonstrated the new functionality and usability provided via hypertext.

2.2.6 ACEC & AES Merger Technical Approach - Boeing

Mr. Kermit Terrell presented a proposed approach to merging the ACEC and AES. As background information he provided a set of high level merged-suite requirements along with a list of technical issues that would need to be addressed prior to the merger. Mr. Terrell also presented details of AES technology and capabilities which should or should not be considered for inclusion in the merged product.

This presentation formed the basis for the issues, recommendations, and discussions that are contained in the following sections. Therefore, these items are not repeated in this section.

2.3 Issues and Recommendations

This section outlines the issues and recommendations that were formulated during the discussion portion of the Workshop. The participants were encouraged to raise issues, provide comments and tender recommendations as if all issues could be researched and all non-conflicting recommendations could be implemented. This approach proved successful in creating a robust list of issues and recommendations, even though the participants knew a priori that the scope of the merger could not support this assumption.

The following issues were raised at the workshop and are briefly discussed in the sections below. Additional discussion on each is provided in the appendices.

- Basic approach for merger
- Development of a merger requirements document

- Utilization of AES performance tests
- Utilization of AES assessors
    - Compiler-related
    - Non-Compiler-related
- Functionality of user interface.
- Ease of test suite setup.
- Functionality of the reporting tools.
- Functionality of the analysis tools

2.3.1 Issue - Basic Merger Approach

Should the merged suite be an adaptation by the ACEC of AES functionality or an adaptation by the AES of ACEC functionality ?

Discussion: The basic approach to merging the two suites should be based upon high level requirements that consider the following: portability, usability, completeness, ease of adaptation, number of users of each suite, and the types of intended users.

The designs of the ACEC and AES were significantly influenced by the developers' understanding of who the end users were intended to be. The AES, which was designed for use on a single host by a centralized test facility, is not easily ported to new host/target combinations and currently has few users. The ACEC, however, which was designed to be portable to accommodate the independent tester, currently has two hundred users who test compilers on many different host and target combinations.

Both suites require an amount of adaptation by the user. The ACEC was designed for ease of adaptation and support is provided to the user in the documentation for this process. The AES emphasized ease of use over portability. For this reason the adaptation effort is higher.

A review of the AES test harness code reveals that it would be difficult and expensive to port this system to hosts beyond the original. Porting this harness to many hosts would not only be cost prohibitive from a development perspective, but from a maintenance one as well.

Another consideration is the user base of each suite. For the hundreds of ACEC users to employ the AES method of doing business, would require a significant investment in learning a new system. This is neither logical nor efficient as there are considerably more ACEC users than AES users.

Recommendation: The merged suite should be an adaptation by the ACEC of AES technology and functionality.

2.3.2 Issue - Development of a Merger Requirements Document

Should a Merger Requirements Document be produced prior to initiation of the merger ?

Discussion: The proposed approach to the merger is a simple blending of ACEC and AES technology and functionality, without the addition of new technology or functionality. That is, if the technology and functionality does not exist in either of the suites, it will not exist in the merged suite.

The ACEC and AES differ significantly in both technology and functionality as a result of having development requirements that varied significantly. However, a union of these requirements, brought about by a simple merger, may not allow the merged suite to meet today's user requirements. The ACEC, for example, has not implemented each and every item on its pre-planned product improvement list. Items such as compiler reliability, which neither suite has addressed, is a good candidate for implementation in the merged suite. There are many more examples of desired technology and functionality for the merged suite which were recommended by the workshop participants.

Recommendation: Although a formal requirements document for the merged suite does not need to be developed, any established requirements should not be just a simple blending of ACEC and AES requirements. The merged suite requirements should allow for technology and functionality which currently does not exist in either suite.

2.3.3 Issue - AES Performance Tests

Should the AES performance tests be included in the merged suite ?

Discussion: The AES contains many tests which will be useful in the merged product as they address technical nuances not addressed by the ACEC. In areas that the ACEC does address, some AES tests are different enough to provide additional useful information. A review of the AES tests has indicated that they should all be thoroughly reviewed before inclusion in the merged suite.

Recommendation: Incorporate AES performance tests into the merged suite after a thorough review and modification as necessary. Where appropriate include in existing ACEC groups and subgroups. If necessary, create new groups and subgroups.

2.3.4 Issue - AES Assessors

Should the AES assessors be included in the merged suite ?

Discussion: The assessors which are common between the two suites are in the areas of capacity limits, diagnostic messages, program library manager and symbolic debugger. These are all compiler-related functions. The AES contains assessors for functions which the ACEC does not. These assessors evaluate both compiler-related and non-compiler-related functions.

There is value in providing the merged suite user with additional assessors which evaluate compiler-related functions. Although these tests are not usually automatable they do provide additional information upon which to base a selection decision.

Recommendation: Incorporate AES compiler-related assessors into the merged suite after a thorough review to eliminate redundancy with ACEC assessors and perform modification as necessary.

2.3.4.1 Issue - Compiler-related Assessors

Which assessors from the AES should be incorporated in the merged suite and in which priority ?

Discussion: The following AES assessors were recommended for inclusion in the merged suite and are listed in priority order as determined by the workshop participants.

- Profiler

- Cross-referencer
- Test coverage analysis
- Test bed generator
- Pretty printing
- Stub generator
- Syntax-based editing
- Assertion checker
- Name expander

Recommendation: Include the assessors named above in the merged suite.

2.3.4.2 Issue - Non-Compiler-related Assessors

Should the merged suite provide assessors that evaluate non-compiler-related tools ?

Discussion: The AES assessors address both compiler and non-compiler-related tools. The ACEC has restricted itself to compiler-related issues since non-compiler evaluation issues were relegated to the Evaluation & Validation Reference System; like the ACEC, a product of the E&V Project.

The increasing size of the test suite is also a concern. Adding non-compiler-related assessors to the suite is an unwarranted and unnecessary growth when these assessors have a natural home in the E&V Reference System.

Recommendation: Incorporate the non-compiler-related assessors from the AES into the E&V Reference System.

2.3.5 Issue - User Interface

Should the merged suite provide all current AES test harness functionality ?

Discussion: The AES test harness provides the user functionality that is not provided by the ACEC. These functions include an interactively accessible database, a command file generation capability, an Ada program generation capability and test-harness-level direct execution mode.

Inspection of the AES source code for the desired functions reveals that many features are host dependent and, therefore, may not be readily ported to other hosts. One reason may be that the operating system for a new host may not be able to provide the same support to implement the desired functions as the original AES host did.

Recommendation: Investigate methods for providing the desired functions from the AES test harness in a portable fashion. Incorporate these functions if they can be implemented in a portable manner.

2.3.5.1 Issue - Interactively Accessible Database

Should the merged suite provide an interactive capability to determine the number of tests that have and have not run, and the tests' status ?

Discussion: This capability would be extremely useful for the user who traditionally runs custom subsets of the tests.

The AES provides a capability to interactively determine which tests have been run and their status. However, the data it outputs are rather cryptic and sometimes difficult to correctly interpret. The ACEC provides this information on its test reports, after completion of testing. Although the data required is available in the ACEC, no ACEC mechanism exists to access that data interactively during the testing process.

Recommendation: Provide the merged suite user with the capability to interactively determine the number of tests that have and have not run, and the tests' status. Provide this capability through existing ACEC data structures if it does not significantly expand the database.

2.3.5.2 Issue - Command File Generation Capability

Should the merged suite provide a command file generation capability for the purpose of implementing a highly interactive test selection user interface ?

Discussion: The AES allows the user to interactively select individual tests for execution via a command file generation capability. The ACEC allows the users to select pre-defined fixed groups of tests by either editing the existing command files or creating new command files, depending on the execution host. However, many users will be interested in running tests which may be combinations of subsets of larger pre-defined groupings. A powerful, highly interactive test selection user interface could provide the functionality required by a user who desires to create custom test groupings.

The solution discussed may require a database capability whose development cost may be far beyond the scope of the merger effort. Additionally, this capability may not be portable which is in direct conflict with the portability requirement.

A compromise solution would be to provide selection capabilities on pre-defined subgroups instead of on individual tests. This approach may alleviate the requirement for the database capability.

Another approach requires additional functionality in the report generation tools. Although this does not solve the selection problem, it does produce only results for desired tests by allowing the user more flexibility in data output selection.

Recommendation: The merged suite should provide the capability to select custom sets of tests for execution, minimally at the ACEC subgroup level. The approach used must be highly portable and as such, shall avoid all non-portable database schemes.

2.3.5.3 Issue - Ada Program Generation Capability

Should the merged suite provide an Ada program generation capability to generate test code ?

Discussion: The AES provides an Ada program generation capability to create test code. This capability is useful when testing a single system or when code is required to determine compiler capacity limits. However, the most common usage of the merged suite will be to compare multiple versions of a compiler or to compare different compilers. In this mode, code that is automatically generated may not be at a level of detail capable of distinguishing between systems. Also, there is some question of repeatability, i.e. whether the same code will be generated each time precisely the same for each system under test.

As for code generation for testing compiler capacity limits, the ACEC already contains a successful mechanism for providing this capability.

Recommendation: Automatic code generation capabilities are of limited value except for capacity testing. This value does not justify an investment in the the merger effort.

2.3.5.4 Issue - Test-Harness-Level Direct Execution Mode

Should the merged suite provide a capability to execute tests without exiting the test harness ?

Discussion: The AES provides an interactive mechanism to select individual tests for execution from the test harness level. The ACEC does not provide this mechanism.

The AES capability is highly dependent upon the VAX operating system utility, STARLET. This utility provides the operating system interface for the user. Although a harness level test selection capability is useful, it does not provide enough utility to justify an attempt to create a portable capability. To begin with, the same capability can be produced in a portable fashion which simply requires the user to temporarily exit the harness to execute the necessary command files. Second, no assumptions can be made about the availability of a STARLET-like utility on any other operating systems. Therefore, if they did not exist they would have to be created by ACEC users for each compilation system, significantly increasing the effort required to adapt the test suite.

Recommendation: Do not implement a harness-level direct execution mode in the merged suite.

2.3.6 Issue - Test Suite Setup

Should ease of setup be a primary requirement ?

Discussion: As the merged suite will be portable and designed to accommodate the individual user, and not a large government-run test facility, great care must be taken in developing an appropriate set-up process. Consideration must be given to the fact that the user's host and target combinations are numerous as will be their experience level in using compiler evaluation suites. It is therefore essential that the user be provided with considerable written assistance for the purpose of initiating the evaluation process. Although this initiation requires the

accomplishment of a limited number of steps prior to actually executing the test suite, if they are not accomplished then the suite cannot be successfully executed.

An acceptable set-up process should meet the following requirements:

- Consists of quality documentation
  - Enumerates the depth of knowledge required by the tester.
  - Identifies key milestones.
  - Completely and unambiguously defines the setup process and procedures.
- Considers the variability of compilers.
- Provides a logical approach to the problem.

Recommendation: Use the ACEC set-up process as a framework for developing the merged suite set-up process and ensure that it meets the requirements listed above.

2.3.7 Issue - Report Styles

Should the analysis reports favor the management-level reader or the evaluation expert-level reader ?

Discussion: The AES produces one type of report. This report is used to document results for a single system and is aimed at the management-level reader. The ACEC produces two types of reports. One is used to document strengths and weaknesses in a single system, while the other is used to compare results from multiple systems. Both are aimed at the compiler evaluator-level reader.

The advantage of a management level report is that conclusions are drawn for the reader who does not have to do any analysis. The disadvantage is the reader is given little if any opportunity to question the conclusions, examine the results, and draw one's own conclusions. The evaluator-level reports provide significant amounts of data and require the reader to understand the technical issues and the process involved in reaching conclusions.

There is a need for both types of reports, one for management and one for the evaluator. As requests for evaluation results by procuring

agencies become more commonplace, the need for the former type of document will increase accordingly. These agencies are not expected to retain compiler evaluation experts who are capable of interpreting evaluator-level reports. As the number of new compilers continues to increase the need for an evaluator-level report will increase also. The anticipated increase in the number of new compilers is a result of anticipated changes to the Ada language via the Ada9X language revision project.

Recommendation: Provide a configurable analysis report capability which selectively provides for the needs of both management personnel and the compiler evaluators.

2.3.8 Issue - Assessor Report Capabilities

Should the merged suite provide the capability to perform comparative analysis of assessors ?

Discussion: Any quantification of data will provide a management level reader with an analysis report they usually seek to avoid doing personally. On the other hand, where the quantified data is qualitative in nature, the reader will be done a disservice in drawings conclusions from this data.

Since the ACEC provides a comparative analysis capability for the performance tests, many users will expect the merged suite to provide the same type of capabilities for the assessor results. As a minimum, results from each system should be output next to each other in a columnar fashion to permit easy, manual comparison of the results

Recommendation: Investigate comparative analysis of assessor results for the merged suite to determine the utility of this capability. If it proves worthwhile, implement this capability in the merged suite.

APPENDIX A - ATTENDEES

ACEC/AES Merger Workshop
FCDSSA, San Diego
22-24 January 1992

NAME	MAILING /NET ADDRESS	TELEPHONE
Merger Workshop Co-chairs:		
SZYMANSKI, Raymond	WL/AAAF-3 WPAFB, OH 45433-6543 szymansk@ajpo.sei.cmu.edu	(513) 255-3947
ROY, Dan	Software Engineering Institute CMU Pittsburgh, PA 15213 dmr@sei.cmu.edu	(412) 268-6180
Merger Workshop Participants		
ASHBY, Sam	Boeing Defense & Space Group P.O. Box 7730, MS K80-13 Wichita, KS 67277-7730 tleavitt@ajpo.sei.cmu.edu	(316)526-2691
ALTMAN, Neal	SEI CMU Pittsburgh, PA 15213 na@sei.cmu.edu	(412) 268-7613
BOWLES, Ken	Telesoft 13040 Caminito Mar Villa Del Mar, CA 92014 kbowles@ajpo.sei.cmu.edu	(619) 755-7288
CRAWFORD, Bard	TASC 55 Walkers Brook Dr. Reading, MA 01867 crawford@ajpo.sei.cmu.edu	(617) 942-2000
DECKER-LINDSEY, Barbara	Boeing Defense & Space Group P.O. Box 7730, MS K80-13 Wichita, KS 67277-7730 tleavitt@ajpo.sei.cmu.edu	316) 523-1500
DONOHOE , Pat	SEI CMU Pittsburgh, PA 15213 pd@sei.cmu.edu	(412) 268-7616

EILERS, Dan	Irvine Compiler 34 Exec Pk, #270 Irvine, CA deilers@ajpo.sel.cmn.edu	(714) 250-1366
EVANS, Bobby	DoD Ada Validation Facility WPAFB, OH 45433 evansbr@adawc.wpafb.af.mil	(513) 255-4472
FERGUSON, Clarence 'Jay"	N.S.A. 958 School Lane Gambrills, MD 21054 cferguson@dockmaster.ncsc.mil	(410) 688-7636
GICCA, Greg	Telesoft 10 Northwood Dr. Merrimack, NH 03054 giccag@ajpo.sei.cmu.edu	(617) 270-0676
GRUBE, Trudy	ORI 3578 Kettering Blvd. Dayton, OH 45439	(513) 299-4141
LANGDON, Major Kim	CECOM Signals Warfare Vint Hill Farms Station Warrenton, VA 22186 langdonk@ajpo.sei.cmu.edu	(703) 349-6938
LEAVITT, Tom	Boeing Defense & Space Group P.O. Box 7730, MS K80-13 Wichita, KS 67277-7730 tleavitt@ajpo.sei.cmu.edu	(316) 523-2023
MCKEE, Gary	McKee Consulting P.O. Box 3009 Littleton, CO 80161 gmckee@ajpo.sei.cmu.edu	(303) 795-7287
RHOADS, Barbara	ORI 3578 Kettering Blvd. Dayton, OH 45439 rhoadsb@ajpo.sei.cmu.edu	(513) 253-2623
TERRELL, Kermit	Boeing Defense & Space Group P.O. Box 7730, MS K80-13 Wichita, KS 67277-7730 leavitt@ajpo.sei.cmu.edu	(316) 523-2022
WOOD, Jon	Institute for Defense Analysis 1801 N. Beauregard Alexandria, VA 22311 wood@ida.org	(703) 845-6632

APPENDIX B - AGENDA

ACEC/AES Merger
FCDSSA, San Diego
22-24 January 1992

Wednesday, 22 January 1992

0800-0830	Visitor and Parking Pass Acquisition at Visitor Control
0830-0845	Orientation/Introduction to FCDSSA Mr. Lloyd Stiles
0845-0900	Co-Chair Comments Dan Roy -- SEI Raymond Szymanski -- Wright Laboratory
0900-1000	ACEC Version 3.0 Boeing
1015-1130	ACEC Version 3.0 (CON'T)
1300-1500	AES Version 2.0 (CANCELLED: UNABLE TO ATTEND) UK MOD
1515-1600	AES Version 2.0 Boeing
1600-1700	ACEC / AES Experience at SEI SEI

Thursday, 23 January 1992

0800-0900	E&V Reference System Version 3.1 - Hypertext based Demonstration TASC
0900-1000	Proposed Technical Approach to Merging the ACEC and AES Boeing
1015-1130	Proposed Technical Approach to Merging the ACEC and AES (CON'T) Boeing
1300-1700	Discussion of Proposed Technical Approach

Friday, 24 January 1992

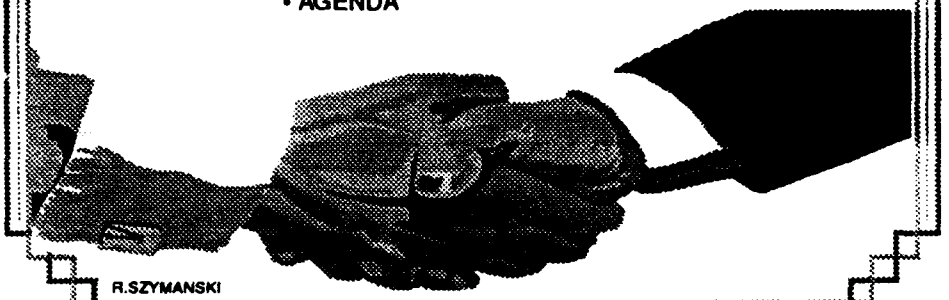
0830-1600	Discussion of Proposed Technical Approach (CON'T)

APPENDIX C - PRESENTATIONS

ACEC/AES
MERGER WORKSHOP

CO-CHAIR COMMENTS

- WORKSHOP OBJECTIVE
- PARTICIPANTS
- WORKSHOP REPORT
- AGENDA



R.SZYMANSKI

WORKSHOP OBJECTIVE

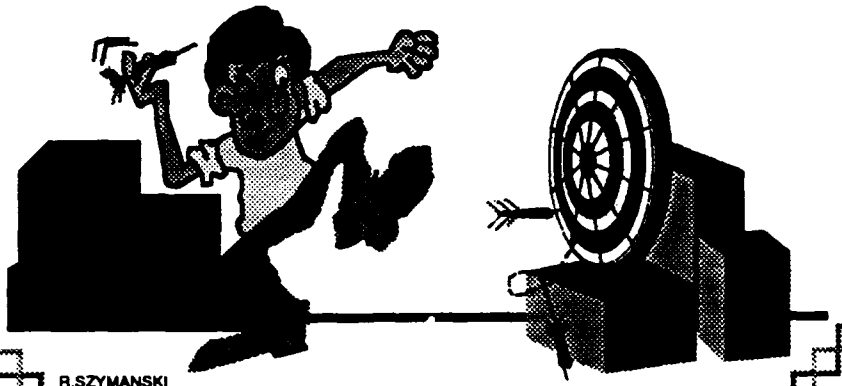
- REVIEW AND REFINE APPROACH TO MERGING ACEC AND AES

ACEC / AES

R.SZYMANSKI

PARTICIPANTS

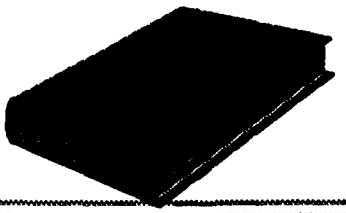
- INVITED ON FOLLOWING BASES
  - ABILITY TO CONTRIBUTE
  - INDIVIDUAL PERSPECTIVE
  - CAPABILITY AS A TEAM PLAYER



R.SZYMANSKI

WORKSHOP REPORT

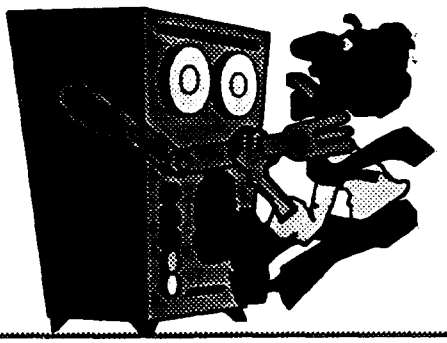
- PRELIMINARY REPORT
  - MAILED 2 WEEKS FOLLOWING WORKSHOP
- FINAL REPORT DRAFT
  - MAILED 4 WEEKS FOLLOWING WORKSHOP
- PROCEEDINGS TO BECOME FORMAL WL TECH REPORT



R.SZYMANSKI

AGENDA

- **CHANGES**
- **HIDDEN**



R.SZYMANSKI

Ada Compiler Evaluation Capability (ACEC)

Contract Number
F33615-86-C-1059
WL/AAAF

Boeing Defense and Space Group
Product Support Division

Where we are

- **Version 3.0 Testing completed**
  - **Ran on 5 systems**
  - **Identified and resolved 205 problems**
- **Version 3.0 to the customer 18 Dec 91**
  - **Tapes (3134 files - 24 megabytes)**
  - **Documentation (printed and on-line)**
- **Under final customer review**
- **Maintenance**

Release Contents

- Total Performance Tests (1627)
  - Groups
  - Subgroups
- Assessors
- Analysis Tools
- User Documentation

Documentation

- User's Guide (352 pages)
  - How to set-up and run
- Reader's Guide (189 pages)
  - Why and how to interpret
- Version Description Doc (398 pages)
  - Descriptions and references

Overview of Release 3.0

Structure

Performance Tests

Assessors

Pretest

Gathering Data

Analysis Tools

Documentation

GROUP AND SUBGROUP ORGANIZATION

Examples from the avionics subgroup of the application group

ap_avdum.ada	– dummies
ap_avpkg.ada	– common packages
ap_av01_.inc	– individual tests
ap_av02_.inc	.
ap_av03_.inc	.
ap_avm01.inc	– first main program
ap_av04_.inc	– individual tests
ap_av05_.inc	.
ap_av06_.inc	.
ap_avm02.inc	– second main program
ap.com	– VMS command file for this group
ap.unx	– UNIX command file for this group

EXECUTION-TIME TEST GROUPS

Group Name	Abbrev	Number of subgroups
Application	ap_	15
Arithmetic	ar_	14
Classical	cl_	13
Data Storage	do_	5
Data Structures	dr_	19
Delays and Timing	dt_	3
Exception Handling	xh_	8
Generics	gn_	2
Input/Output (MIS)	io_	12
Miscellaneous	ms_	6
Optimizations	op_	30
Program Organization	po_	4
Statements	st_	10
Storage Reclamation	sr_	2
Subprograms	su_	8
Systematic Compile Speed	sy_	11
Tasking	tk_	7

ASSESSOR GROUPS

Group Name	Abbrev	Subgroup Name	Abbrev
Capacity Assessor	yc_	compile_time	ct
		run_time	rt
Debugger Assessor	yb_		
Diagnostics Assessor	yd_	compiler_errors	ce
		compiler_warnings	cw
		link_time	lt
		run_time	rt
Library Assessor	yl_		

SUPPORT GROUPS

Group Name	Abbrev	Subgroup Name	Abbrev
Analysis	za_	Comparative Analysis	ca
		Condense	cn
		Menus	mn
		Single System Analysis	sa
Command Files	zc_		
Documentation	zd_		
Global & Timing Files	zg_		
Math Packages	zm_		
Pretest	zp_		

COMMAND FILES

- Organized by group & subgroup
- Standardized command spelling
- Glossary of commands
- Ada programs for difficult-to-adapt steps
  - *Compilation time stamps, calculations*
  - Capacity test calculations

Overview of Release 3.0

- Structure
- Performance Tests
- Assessors
- Pretest
- Gathering Data
- Analysis Tools
- Documentation

Performance Tests

Execution Time	1627 tests
Code Size	1627 tests
Compile Speed	588 tests
Link Speed	571 tests
Compile and Link Speed	588 tests
Systematic Compile Speed Group	92 tests

SYSTEMATIC COMPILE SPEED

Test Area (subgroups)	Number
compilation_unit_size	45
compile_time_arithmetic	2
generics	12
optimization	6
pragma_inline	3
program_library_size	2
smart_recompilation	6
source_presentation	4
subunits	6
symbol_table_size	2
with_clauses	4

NEW PERFORMANCE TESTS

Test Area	Number
Array of records vs record of arrays vs parallel arrays	3
Zero vs non-zero based arrays	4
Coding style: CASE vs IF	4
Coding style: exception raising vs explicit IF	4
Reclamation test using function returning an unconstrained type in several contexts	1
Algebraic simplification "handedness" bias	3
Allocate statically sized storage in blocks	4
UNCHECKED_CONVERSION between arrays and records	4
Passing integer parameters	12
"+" and "-" functions for TIME and DURATION	4
Reordering expressions	8
High-precision temporaries	6

NEW PERFORMANCE TESTS CONTINUED

Test Area	Number
SELECT with variable number of ACCEPT alternatives	4
Algorithm used in selective wait	2
Order of evaluation of guards in a selective wait	1
Variability of exception processing time with number of tasks	2
Reclamation test for task created via allocation	1
Variability of task creation with a pre-existing active tasks	2
Task-switch time	3
Scheduling of task or master on creation	1
Task scheduling after interrupt	1
A rule-based expert system	1
Caching/paging	36
Pipelining	8
Shared variable	2

RUN-TIME MEMORY SIZE

- Determine size of run-time objects
- Write as performance tests with ancillary data
- Use 'SIZE where possible, otherwise 'ADDRESS
- Variability with respect to common optimizations
- Structures to measure
  - Task control blocks
  - Activation records
  - Variant records
  - Objects of an unconstrained type

Overview of Release 3.0

- Structure
- Performance Tests
- Assessors
- Pretest
- Gathering Data
- Analysis Tools
- Documentation

ASSESSORS

- Diagnostic Assessor
- Debugger Assessor
- Library Assessor
- Capacity Assessor -- New

For each assessor

- Readme file
- Report template

DIAGNOSTIC ASSESSOR

Tests:

- Compiler error messages - 34 tests
- Compiler warning messages - 30 tests
- Link-time error messages - 7 tests
- Run-time error messages - 10 tests

DIAGNOSTIC ASSESSOR

Template questions:

- Is the diagnostic message printed?
- Is the message in the general area of difficulty?
- Is the message in the correct specific location?
- Does the text of the message clearly define the difficulty?
- Is relevant non-local information listed where appropriate?
- Is error recovery appropriate?

DEBUGGER ASSESSOR

29 debugging scenarios
User performs debugging operations
New:
- Labels
- Line numbered files

Tests:
- Functional capabilities
- Performance
- Capacity

LIBRARY ASSESSOR

22 scenarios
New:
- Times, sizes collected in Systematic Compile
  - Speed group

Tests:
- Functional capabilities
- Performance
- Capacity

CAPACITY ASSESSOR

Compile-time tests - 32
Run-time tests - 9

Testing guided by:
- default or user-selected ranges of values
- default or user-selected time limit

Branch-and-bound plus binary search technique

COMPILE-TIME TESTS

Source code generated at time of test by supplied source generators

Tests static limits definable at compile time:
- Quantity -- names, tasks, elements, etc.
- Size -- literal pool, declarative region
- Depth of nesting -- IF, generics, subunits, etc.

RUN-TIME TESTS

May result in system crash on weaker systems

Tests dynamic features defined at run time:

- Quantity -- tasks, objects, elements, etc.
- Size -- arrays, collection, data segment
- Depth of nesting -- subprogram calling

Overview of Release 3.0

Structure

Performance Tests

Assessors.

Pretest

Gathering Data

Analysis Tools

Documentation

PRETEST

Purpose

- Aid user in getting started
- Organize adaptation effort
- Provide useful system information
- Prepare to execute test suite, analyze data

Contents

- Readme file – zp_rdme1.txt
- Test programs, command files
- Report template

PRETEST CONTINUED

1. Access Ada Compilation System
2. Test label'ADDRESS
3-4. System Clock/Calendar Tests
5. Compile Baseline Files *{Mandatory}*
6. Test Inner Timing Loop Iteration Count
7-9. Test Math Package Adaptation
10. Test Preprocessor zg_incld
11. Test Performance Command Files *{Mandatory}*
12. Compile Analysis Tools
13. Test Condense
14. Test Comparative Analysis

Overview of Release 3.0

- Structure
- Performance Tests
- Assessors
- Pretest
- Gathering Data
- Analysis Tools
- Documentation

COLLECTION OF RESULTS

- Test results written to standard output
  - Compilation log (host)
  - Execution log (target)
- Save logs to text files
- Input to analysis phase

EXECUTION LOG

```
\ACEC begin mainprogram\**********AR_FL_M01
                                          outer loop count
                                    inner loop count      |
                              bits  microseconds |        |
problem name                  size  min    mean  |    |   sigma
\acec_problem_name\ ar_fl_flt_oper_01
    xx:= 1.0;
\acec_measurements\            96   43.7   45.1  15   3   3.5%#
\acec_problem_name\ ar_fl_flt_oper_02
    xx:= yy;
\acec_measurements\           128  111.3  114.2  14   4   2.9%
>>> ancillary data

\ACEC end   mainprogram\**********AR_FL_M01
```


COMPILE AND LINK TIMES

Ada programs bracket commands, issue time stamps

Elapsed, CPU versions

Calculations performed in CONDENSE

- Subtract overhead
- Error checking

```
\acec begin\  AP_AVM01
\acec begin el\  2380.000  21 DEC 1991
\acec end  el\  2400.000  21 DEC 1991
\acec end\  AP_AVM01
```

Overview of Release 3.0

- Structure
- Performance Tests
- Assessors
- Pretest
- Gathering Data
- Analysis Tools
- Documentation

Analysis Tools

- Menu
- Condense
- Comparative Analysis
- Single System Analysis

ANALYSIS MENU

- Portable interface to
  - Condense
  - Comparative Analysis
  - Single System Analysis
- Link with 0-3 tools, depending on space
  - Dummies provided
- Analysis tools executable
  - From menu
  - Batch, using request files created by menu

CONDENSE

- Call from Menu, SSA, CA, or batch
- Convert log files to database files
  - Run-time error diagnosis
  - Compute compilation times
  - Cross check execution, compilation results
- Incremental mode adds to database
- Optional reports
  - No Data Report
  - Exceptional Data Report
  - Multiple Data Report

DATABASES

- Execution time/Code size
- Compilation time/Link time
- Text files
  - Readable, modifiable by user
  - By group, subgroup
  - Duplicate results adjacent
  - One result selected for analysis
- Input to Comparative, Single System Analysis

Comparative Analysis

- Groups
- Summary of all groups
- Application profile mode
- System factors & confidence intervals
- Outliers

Main Menu

Main Menu

a. -CONDENSE
b. -COMPARATIVE_ANALYSIS
c. -SINGLE SYSTEM_ANALYSIS

HElp QUit NExt

Select 1 tool (separate selections with comma):

=> "b,ne" <cr>

Work Shop BOEING 1/16/92 39

System Menu

System Menu

a. -system_1
b. -system_2
c. -system_3
d. -All Systems

HElp QUit MAin
NExt PRevious

Select 2 or more systems to be compared (separate selections with comma):

=> "d,ne" <cr>

Work Shop BOEING 1/16/92 40

Metrics Menu

```
Metrics Menu
a. -EXECUTION_TIME                          := .tim
b. -CODE_SIZE                               := .siz
c. -COMPILATION_TIME                        := .cmp
d. -LINK_TIME                               := .lnk
e. -COMBINED COMPILATION_LINK_TIME          := .cml
f. -All Metrics

HElp     QUit     MAin
NExt     PRevious DEfault names

Select 1 or more metrics (separate selections with comma):
=> "a,c,ne" <cr>
```


Groups Menu

```
Groups Menu
a. -APPLICATION                             := applic00
b. -ARITHMETIC                              := arithm00
c. -CLASSICAL                               := classi00
d. -DATA_STORAGE                            := storag00
e. -DATA_STRUCTURES                         := struct00
f. -DELAYS_AND_TIMING                       := delays00
g. -EXCEPTION_HANDLING                      := except00
h. -GENERICS                                := generi00
i. -INPUT_OUTPUT                            := input_00
j. -MISCELLANEOUS                           := miscel00
k. -OPTIMIZATIONS                           := optimi00
l. -PROGRAM_ORGANIZATION                    := progra00
m.-STATEMENTS                               := statem00
n. -STORAGE_RECLAMATION                     := reclam00
o. -SUBPROGRAMS                             := subpro00
p. -SYSTEMATIC_COMPILE_SPEED                := system00
q. -TASKING                                 := taskin00
r. -All Groups
```

CA Report Options

```
------------------------ CA Report Options ------------------------
a. -SUMMARY_REPORT
b. -FULL_REPORT and SUMMARY_REPORT
c. -SUMMARY_OF_ALL_GROUPS_REPORT              := summry00
d. -All Above
e. -SPECIAL_REPORT                            := specia00
f. -Write all reports in current request to one file   := compar00.rpt
g. -Change length of output line to           := 80
-------------------------------------------------------------------

HElp      QUit      MAin
NExt      PRevious  DEfault names

Select 1 or more reports (separate selections with comma):
=> "b,f,ne" <cr>
```


Run or Save Request

```
Current Selection is
PROGRAM :   COMPARATIVE_ANALYSIS
SYSTEMS  :  system_1
            system_2
            system_3
METRICS  :  EXECUTION_TIME, COMPILATION_TIME
GROUPS   :  DATA_STORAGE, STATEMENTS, TASKING
OPTIONS  :  SUMMARY_REPORT, FULL_REPORT,
            One output file.
            Output line length: 80
a. -Run immediately
b. -Store request in new Request file
c. -Append request to existing Request file
-------------------------------------------------------------------
HElp      QUit      MAin      PRevious      DO request
Select 1 option, and enter "DO" to apply (separate selections with comma):
=> "a,do" <cr>
```

Single System Analysis

- Language Feature Overhead
- Optimizations
- Coding Style Variations
- Ancillary Data
- Failure Analysis
- Compile Speed Analysis
- Code Size Analysis

Overview of Release 3.0

- Structure
- Performance Tests
- Assessors
- Pretest
- Gathering Data
- Analysis Tools
- Documentation

USER'S GUIDE

- Pretest directions
- Readme files
- Trouble shooting guide
- Glossary of commands
- Steps for adding tests
- Non-generic version of MATH
- Running on a simulator
- Groups and subsets of tests
- Changing compilation options
- Modifying tests to use system-dependent features

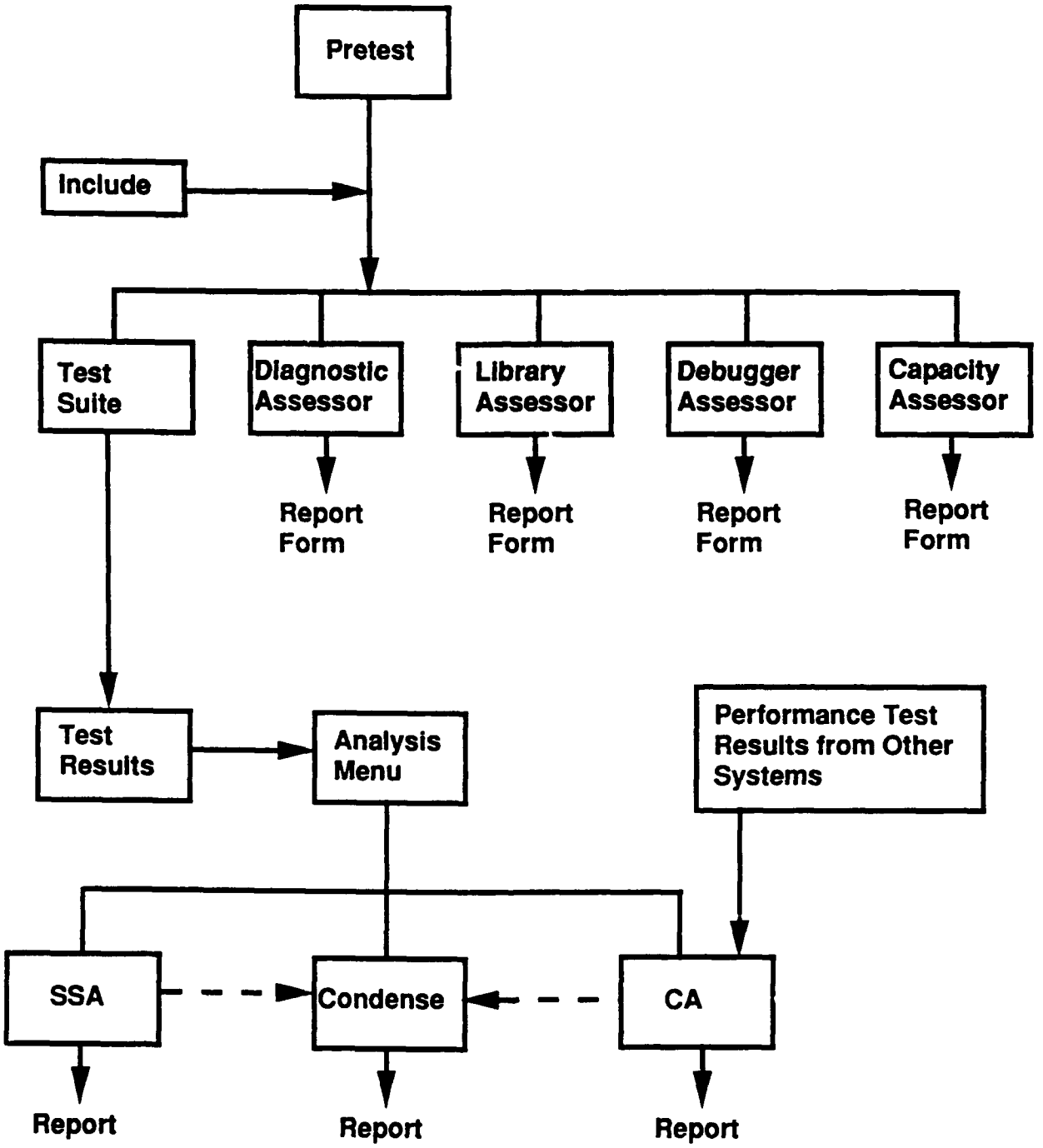
READER'S GUIDE

- Organization of the test suite
- Citations to other works
- Report reviews
- Interpretation the analysis reports
- Interpreting tests with system-dependent modifications
- Interpretation of compilation time results
- Implementation trade-offs

VERSION DESCRIPTION DOCUMENT

- Test problem descriptions
- Test problem to source file map
- Tape description
- ACEC keyword indexes
- Quarantined test problems
- Mapping of old to new names
- System dependent test problems
- Optimization techniques
- Assessor information

THE END



Overview of the ACEC Version 3

Code Size Report

```
Code Size Report : (physical lines)

----- average bytes per lines :      10.69
   -- based on total line count of      20160
-----
----- HIGHEST           -- Test : io_tx_io_09
   -- bytes per line      :    757.00      -- line count      :      1
   -- bytes per semicolon:    757.00      -- semicolon count:      1
-----
----- LOWEST            -- Test : sr_ex_explicit_04
   -- bytes per line      :      0.06      -- line count      :     84
   -- bytes per semicolon:      0.17      -- semicolon count:     29
-----

Code Size Report : (semicolons)

----- average bytes per semicolons :      16.23
   -- based on total semicolon count of      13281
-----
----- HIGHEST           -- Test : io_tx_io_09
   -- bytes per line      :    757.00      -- line count      :      1
   -- bytes per semicolon:    757.00      -- semicolon count:      1
-----
----- LOWEST            -- Test : po_pa_d_library_05
   -- bytes per line      :      0.10      -- line count      :     48
   -- bytes per semicolon:      0.14      -- semicolon count:     35
-----

Code Size Report : (Examples)

-----
      -- Test : cl_dh_dhrys_01
   -- bytes per line      :     10.32      -- line count      :     47
   -- bytes per semicolon:     19.40      -- semicolon count:     25
-----
      -- Test : cl_dh_dhrys_02
   -- bytes per line      :      7.00      -- line count      :     47
   -- bytes per semicolon:     13.16      -- semicolon count:     25
-----
      -- Test : cl_dh_dhrys_03
   -- bytes per line      :      7.00      -- line count      :     47
   -- bytes per semicolon:     13.16      -- semicolon count:     25
-----
      -- Test : cl_wh_whet_01
   -- bytes per line      :     29.13      -- line count      :    134
   -- bytes per semicolon:     44.36      -- semicolon count:     88
-----
```

Compile Speed Report

Compilation Speed : (physical lines)

----- average lines per minute : 111.12
-- based on total line count of 211908.0

----- HIGHEST -- File : ap_kfm01
-- lines per minute : 1570.75 -- line count : 4762
-- semicolons per minute: 811.76 -- semicolon count: 2461

----- LOWEST -- File : sy_cum21
-- lines per minute : 0.90 -- line count : 209
-- semicolons per minute: 0.69 -- semicolon count: 162

Compilation Speed : (semicolons)

----- average semicolons per minute : 66.66
-- based on total semicolon count of 127120.0

----- HIGHEST -- File : po_msm09
-- lines per minute : 379.35 -- line count : 4348
-- semicolons per minute: 1055.26 -- semicolon count: 12095

----- LOWEST -- File : sy_cum21
-- lines per minute : 0.90 -- line count : 209
-- semicolons per minute: 0.69 -- semicolon count: 162

Compilation Speed : (Examples)

-- File : cl_dhm01
-- lines per minute : 578.57 -- line count : 756
-- semicolons per minute: 280.87 -- semicolon count: 367

-- File : cl_dhm02
-- lines per minute : 615.72 -- line count : 744
-- semicolons per minute: 306.21 -- semicolon count: 370

-- File : cl_dhm03
-- lines per minute : 619.89 -- line count : 748
-- semicolons per minute: 309.12 -- semicolon count: 373

-- File : cl_whm01
-- lines per minute : 306.48 -- line count : 331
-- semicolons per minute: 173.15 -- semicolon count: 187

Failure Analysis Report - Execution Results

Failure Analysis : by Group and by Type of Failure

Groups	Data Summary Categories											
	Valid	CmpT	RunT	noDa	Depn	Pkng	Unrl	XcsT	Dely	Vrfy	Othr	Total
application	73	13	0	0	0	0	0	0	0	0	0	86
arithmetic	108	0	0	0	0	0	4	0	0	0	0	112
classical	83	0	0	0	0	0	0	0	0	0	0	83
data_storag	91	0	0	0	0	0	0	0	0	0	0	91
data_struct	224	0	0	0	0	0	1	0	0	0	0	225
delays_and_	26	0	0	0	0	0	0	0	8	7	0	41
exception_h	58	0	0	0	0	0	0	0	0	0	0	58
generics	24	0	0	0	0	0	0	0	0	0	0	24
input_outpu	105	0	0	0	0	0	3	0	0	0	0	108
miscellaneo	17	0	0	0	0	0	0	0	0	0	0	17
optimizatio	304	0	1	0	0	0	0	0	0	0	0	305
program_org	74	0	0	0	0	0	0	0	0	0	0	74
statements	80	0	0	0	0	0	0	0	0	0	0	80
storage_rec	47	0	0	0	0	0	0	13	0	0	0	60
subprograms	79	0	0	0	0	0	0	0	0	0	0	79
systematic_	75	0	0	0	0	0	0	0	0	0	0	75
tasking	98	1	0	0	1	0	1	0	8	0	0	109
Totals	1566	14	1	0	1	0	9	13	16	7	0	1627

Failure Analysis Report - Compilation Results

Failure Analysis : by Group and by Type of Failure

Groups	Data Summary Categories											
	Valid	CmpT	RunT	noDa	Depn	Pkng	Unrl	XcsT	Dely	Vrfy	Othr	Total
application	35	0	0	2	0	0	0	0	0	0	0	37
arithmetic	22	0	0	0	0	0	0	0	0	0	0	22
classical	38	0	0	0	0	0	0	0	0	0	0	38
data_storag	14	0	0	0	0	0	0	0	0	0	0	14
data_struct	50	0	0	0	0	0	0	0	0	0	0	50
delays_and_	10	0	0	2	0	0	0	0	0	0	0	12
exception_h	17	0	0	0	0	0	0	0	0	0	0	17
generics	5	0	0	0	0	0	0	0	0	0	0	5
input_outpu	22	0	0	1	0	0	0	0	0	0	0	23
miscellaneo	7	0	0	0	0	0	0	0	0	0	0	7
optimizatio	64	0	0	1	0	0	0	0	0	0	0	65
program_org	20	0	0	0	0	0	0	0	0	0	0	20
statements	17	0	0	0	0	0	0	0	0	0	0	17
storage_rec	51	0	0	0	0	0	0	0	0	0	0	51
subprograms	16	0	0	0	0	0	0	0	0	0	0	16
systematic_	91	0	0	1	0	0	0	0	0	0	0	92
tasking	100	0	0	2	0	0	0	0	0	0	0	102
Totals	579	0	0	9	0	0	0	0	0	0	0	588

Ancillary Data

Ancillary Data - List

```
>>> ap_kf_kalman
>>> approximate time per filter call:    4088.5 number of iterations:    800
>>> cl_ac_acker_01
>>> time per call is      12.0
>>> cl_ac_acker_02
>>> time per call is      17.9
>>> cl_dp_task_02
>>> Time per rendezvous =      582.1
>>> cl_dp_task_03
>>> time per rendezvous =      606.6
>>> cl_dp_task_04
>>> time per rendezvous =      581.1
>>> cl_dp_task_05
>>> time per rendezvous =      449.1
>>> cl_dh_dhrys_01
>>> Dhrystones per second, checking:    2553.99
>>> cl_dh_dhrys_02
>>> Dhrystones per second, checking suppressed:    3975.81
>>> cl_dh_dhrys_03
>>> Dhrystones per second, optimize(space),nochecking:  3960.90
>>> dr_ao_array_oper_32
>>> 2-D arrays allocated in row-major order.
>>> dr_ss_tcb_01
>>> Task Control Block size is  256 bits,  32 8-bit bytes
>>> DR_SS_ACTIV_REC_01
>>> activation record size is  416 bits,  52 8-bit bytes
>>> DR_SS_ACTIV_REC_02
>>> activation record size is  352 bits,  44 8-bit bytes
```

Optimizations

Dead Code Elimination

Description		Optimized?
Time : op_de_dead_05 (	6.6) vs	
ar_io_integer_oper_01 (	0.4)	no
Size : op_de_dead_05 (	144.0) vs	
ar_io_integer_oper_01 (	32.0)	

```
op_de_dead_05
        FOR i IN int'(1)..int'(5)
        LOOP    ii := i ;
        END LOOP ;
        ii := 0 ;
    -- dead assignments within loop, killed by assignment
    -- after exit.
    ar_io_integer_oper_01  kk := 1 ;
```

Dead Code Elimination

Description		Optimized?
Time : op_de_dead_06 (	1.0) vs	
st_nu_null_01 (	0.0)	yes
Size : op_de_dead_06 (	48.0) vs	
st_nu_null_01 (	0.0)	

```
op_de_dead_06
      DECLARE
         xyz : real ;
      BEGIN
         xyz := yy * zz ;
      END ;
    -- dead assignments within a block.  Variable assigned to
    -- local which is not referenced before block is exited.
    st_nu_null_01  NULL ;
```

Optimizations

Dead Code Elimination

Description	Optimized?
Time : op_de_dead_02 (1.0) vs ar_io_integer_oper_04 (0.3)	maybe
Size : op_de_dead_02 (80.0) vs ar_io_integer_oper_04 (40.0)	

```
op_de_dead_02  ii := 11 ;  ii := mm ;
-- first assignment is dead
-- Optimization test for dead assignment elimination on integers.
ar_io_integer_oper_04  kk := 11 ;
```

Dead Code Elimination

Description	Optimized?
Time : op_de_dead_03 (1.9) vs ar_fl_flt_oper_02 (1.0)	maybe
Size : op_de_dead_03 (96.0) vs ar_fl_flt_oper_02 (48.0)	

```
op_de_dead_03    xx := yy ;    xx := zz ;
-- first assignment is dead
-- dead assignment elimination; floating point variable
ar_fl_flt_oper_02    xx := yy ;
```

Dead Code Elimination

Description	Optimized?
Time : op_de_dead_04 (0.0) vs st_nu_null_01 (0.0)	yes
Size : op_de_dead_04 (0.0) vs st_nu_null_01 (0.0)	

```
op_de_dead_04  xx := xx ;
-- Assign float variable to itself.
st_nu_null_01  NULL ;
```

Subprogram Calls: With 0..3 Parameters

Test Name	Execution Time	Bar Chart	Similar Groups
su_se_external_01	6.7	*******	
su_se_external_02	8.8	*********	
su_se_external_03	14.9	***************	
su_se_external_04	23.3	************************	

Code Size

su_se_external_01	48.0	***
su_se_external_02	200.0	**********
su_se_external_03	328.0	*****************
su_se_external_04	456.0	************************

Individual Test Descriptions

```
su_se_external_01   proc0 ;
   -- simple procedure with no parameters; call to library scope
   -- procedure : body is null.
```

```
su_se_external_02   proc1 ( xx ) ;
   -- simple procedure with one IN OUT floating point parameter,
   -- declared in external library unit : body is null.
```

```
su_se_external_03   proc2 ( xx , yy ) ;
   -- simple procedure with two IN OUT floating point parameters,
   -- declared in external library unit : body is null.
```

```
su_se_external_04   proc3 ( xx , yy , zz ) ;
   -- simple procedure with three IN OUT floating point parameters,
   -- declared in external library unit : body is null.
```

Language Feature Overhead

Performance Range			Significant Difference?	Missing Tests	Total Tests	Page
Subprogram Calls: With 0..3 Parameters						54
6.7	..	23.3	yes	0	4	

Optimizations

Yes	Maybe	No	Missing	No Stat	Total	Page
Dead Code Elimination						89
11	3	3	0	0	17	

---- Pairwise Comparisons: total n = 17

Systems	v_91	t_91	s_91	m_91	d_91	Mean
n:	17	16	17	17	16	Vari-
Sys Factor:	0.37	1.60	0.30	0.30	2.37	ation
v_91 n:		16	17	17	16	
Sys Factor:		0.37	0.37	0.37	0.37	0.0%
t_91 n:	16		16	16	16	
Sys Factor:	1.60		1.60	1.60	1.60	0.0%
s_91 n:	17	16		17	16	
Sys Factor:	0.30	0.30		0.30	0.30	0.2%
m_91 n:	17	16	17		16	
Sys Factor:	0.30	0.30	0.30		0.30	0.0%
d_91 n:	16	16	16	16		
Sys Factor:	2.37	2.37	2.37	2.37		0.0%

---- Number of Test Problems in the Analysis

No. Problems	v_91	t_91	s_91	m_91	d_91	Possible
application	35	30	33	29	27	37
arithmetic	22	22	22	21	21	22
classical	38	35	38	36	33	38
data_storage	13	11	6	10	10	14
data_structures	50	50	50	49	46	50
delays_and_timing	10	10	9	11	10	12
exception_handling	17	17	17	16	17	[illegible]
generics	5	0	5	4	0	5
input_output	21	18	22	18	4	23
miscellaneous	7	6	7	6	6	7
optimizations	64	65	65	63	64	65
program_organization	20	20	16	18	17	20
statements	17	17	17	17	16	17
storage_reclamation	51	48	41	7	41	51
subprograms	16	16	16	16	16	16
systematic_compile_speed	89	88	80	79	24	92
tasking	90	89	89	87	86	102
Total	565	542	533	487	438	588

Summary over all groups : product data - product model

Raw Data:	v_91	t_91	s_91	m_91	d_91	Wgts
application	0.35	1.32	0.29	0.33	2.60	1.0
arithmetic	0.43	1.91	0.24	0.31	2.08	1.0
classical	0.39	1.63	0.30	0.29	2.47	1.0
data_storage	0.35	1.70	0.37	0.22	1.79	1.0
data_structure	0.35	1.71	0.29	0.29	2.35	1.0
delays_and_tim	0.30	1.50	0.30	0.31	2.69	1.0
exception_hand	0.35	1.48	0.27	0.29	2.53	1.0
generics	1.26		0.81	0.84		1.0
input_output	0.50	2.71	0.32	0.37	2.26	1.0
miscellaneous	0.28	1.38	0.27	0.29	2.74	1.0
optimizations	0.37	1.69	0.33	0.29	2.30	1.0
program_organi	0.31	1.40	0.29	0.29	2.53	1.0
statements	0.41	1.90	0.28	0.29	2.13	1.0
storage_reclam	0.31	1.14	0.29	0.22	2.33	1.0
subprograms	0.40	1.75	0.32	0.32	2.15	1.0
systematic_com	0.62	1.55	0.62	0.45	2.23	1.0
tasking	0.27	1.35	0.30	0.32	2.73	1.0

---- Outlier Statistics: residual * system factor * row mean = actual

	Bounds	Expect	Got	v_91	t_91	s_91	m_91	d_91
-- Very Low :	0.76	2	1	1	0	0	0	0
- Low :	0.80	2	3	2	0	0	0	1
+ High :	1.27	2	0	0	0	0	0	0
++ Very High:	1.33	2	8	2	1	3	2	0
Totals :		8	12	5	1	3	2	1

Residuals	v_91	t_91	s_91	m_91	d_91	Means
applicati	0.98	0.84	0.99	1.14	1.12	0.98
arithmeti	1.19	1.20	0.82	1.03	0.89	0.99
classical	1.05	1.00	1.00	0.95	1.03	1.02
data_stor	1.08	1.20	1.40++	0.83	0.86	0.88
data_stru	0.97	1.07	0.98	0.97	0.99	1.00
delays_an	0.80-	0.92	0.98	1.02	1.12	1.02
exception	0.96	0.94	0.94	0.98	1.09	0.98
generics	3.55++		2.81++	2.91++		0.97
input_out	1.12	1.38++	0.87	1.00	0.77-	1.23
miscellan	0.77-	0.87	0.93	0.98	1.17	0.99
optimizat	1.00	1.06	1.11	1.00	0.98	0.99
program_o	0.89	0.91	1.02	1.02	1.10	0.97
statement	1.11	1.18	0.94	0.99	0.90	1.00
storage_r	0.98	0.83	1.13	0.88	1.15	0.86
subprogra	1.12	1.10	1.09	1.10	0.92	0.99
systemati	1.54++	0.89	1.90++	1.37++	0.86	1.09
tasking	0.74--	0.85	1.02	1.09	1.16	1.00
Sys Fact	0.37	1.60	0.30	0.30	2.37	

Compile and Link --- 20 Dec 1991 12:14:35 --- Page 2

Summary over all groups : product data - product model

---- System Names and Descriptions

v_91
  -- Host: VAXstation 3100 Target: VAXstation 3100
t_91
  -- Host: VAXstation 3100 Target: VAXstation 3100
s_91
  -- Host: MIPS R2000A/R3000 Target: MIPS R2000A/R3000
m_91
  -- Host: DECstation 3100 MIPS RISC Target: DECstation 3100 MIPS RISC
d_91
  -- Host: VAX 6220 Target: 1750A

---- System Factors and Confidence Intervals (including graph)

Systems	Low	Mean	High	Ratio	0.3 ... 2.6
v_91	0.32	0.37	0.42	1.00	|+|
t_91	1.42	1.60	1.80	4.37	|--+--|
s_91	0.27	0.30	0.32	0.81	|+|
m_91	0.28	0.30	0.31	0.81	|+|
d_91	2.14	2.37	2.62	6.47	|--+---|

---- Significant Diff = *

---- Data Summary: Total n = 588

	v_9	t_9	s_9	m_9	d_9	Gps	Valid	NoData	Comp	RunTim	Exclu	Other
v_91		*	*	*	*	17	565	9	0	0	14	0
t_91	*		*	*	*	16	542	43	0	0	3	0
s_91	*	*		-	*	17	533	55	0	0	0	0
m_91	*	*	-		*	17	487	101	0		0	
d_91	*	*	*	*		16	438	148	0	0	2	

---- Group Weights

application	1.0	arithmetic	1.0	classical	1.0
data_storage	1.0	data_structures	1.0	delays_and_timing	1.0
exception_handlin	1.0	generics	1.0	input_output	1.0
miscellaneous	1.0	optimizations	1.0	program_organizat	1.0
statements	1.0	storage_reclamati	1.0	subprograms	1.0
systematic_compil	1.0	tasking	1.0		

Compile and Link --- 20 Dec 1991 12:14:35 --- Page 1

Summary over all groups : product data - product model

---- Pairwise Comparisons: total n = 17

Systems	v_91	t_91	s_91	m_91	d_91	Mean
n:	17	17	15	17	17	Vari-
Sys Factor:	1.08	1.24	0.31	0.58	1.59	ation
v_91 n:		17	15	17	17	
Sys Factor:		1.08	1.09	1.08	1.08	0.2%
t_91 n:	17		15	17	17	
Sys Factor:	1.24		1.24	1.24	1.24	0.0%
s_91 n:	15	15		15	15	
Sys Factor:	0.31	0.31		0.31	0.31	0.0%
m_91 n:	17	17	15		17	
Sys Factor:	0.58	0.58	0.61		0.58	1.1%
d_91 n:	17	17	15	17		
Sys Factor:	1.59	1.59	1.65	1.59		1.0%

---- Number of Test Problems in the Analysis

No. Problems	v_91	t_91	s_91	m_91	d_91	Possible
application	68	76	52	61	40	86
arithmetic	108	111	39	107	108	112
classical	83	80	82	79	54	83
data_storage	91	79	27	75	70	91
data_structures	224	208	135	215	204	225
delays_and_timing	26	16	16	25	26	41
exception_handling	58	52	39	48	39	58
generics	19	19	7	19	15	24
input_output	104	94	55	93	19	108
miscellaneous	16	16	0	16	16	17
optimizations	304	304	168	288	299	305
program_organization	74	74	54	72	5	74
statements	80	80	57	80	77	80
storage_reclamation	47	53	46	50	29	60
subprograms	79	74	36	79	79	79
systematic_compile_speed	73	71	0	59	20	75
tasking	87	89	81	78	75	109
Total	1541	1496	894	1444	1175	1627

Execution Times --- 20 Dec 1991 12:14:16 --- Page 3

Summary over all groups : product data - product model

Raw Data:	v_91	t_91	s_91	m_91	d_91	Wgts
application	1.11	1.35	0.31	0.44	1.64	1.0
arithmetic	0.76	1.06	0.25	0.59	1.42	1.0
classical	0.85	1.11	0.40	0.79	1.91	1.0
data_storage	0.75	0.84	0.27	1.00	1.04	1.0
data_structure	0.88	0.95	0.25	0.65	1.57	1.0
delays_and_tim	0.64	1.11	0.50	0.54	1.29	1.0
exception_hand	1.26	0.82	0.33	0.37	1.18	1.0
generics	0.90	0.99	0.11	0.24	1.86	1.0
input_output	1.05	1.47	0.20	0.35	0.12	1.0
miscellaneous	1.05	1.30		0.43	1.02	1.0
optimizations	0.93	1.44	0.21	0.51	1.23	1.0
program_organi	1.11	1.20	0.28	0.86	1.78	1.0
statements	1.04	1.35	0.19	0.49	1.56	1.0
storage_reclam	1.05	1.20	0.43	0.71	1.49	1.0
subprograms	0.99	0.98	0.16	0.46	1.63	1.0
systematic_com	1.21	1.11		0.46	1.28	1.0
tasking	1.54	0.94	0.51	0.46	1.28	1.0

---- Outlier Statistics: residual * system factor * row mean = actual

	Bounds	Expect	Got	v_91	t_91	s_91	m_91	d_91
-- Very Low :	0.62	2	4	0	0	2	1	1
- Low :	0.67	2	1	0	0	1	0	0
+ High :	1.53	2	0	0	0	0	0	0
++ Very High:	1.65	2	4	0	1	2	1	0
Totals :		8	9	0	1	5	2	1

Residuals	v_91	t_91	s_91	m_91	d_91	Means
applicati	1.06	1.12	1.02	0.77	1.07	0.97
arithmeti	0.86	1.05	0.98	1.25	1.10	0.82
classical	0.78	0.88	1.25	1.35	1.19	1.01
data_stor	0.89	0.87	1.11	2.21++	0.84	0.78
data_stru	0.95	0.89	0.93	1.30	1.15	0.86
delays_an	0.73	1.10	1.94++	1.14	0.99	0.81
exception	1.46	0.84	1.34	0.81	0.94	0.79
generics	1.01	0.98	0.41--	0.51--	1.43	0.82
input_out	1.52	1.86++	0.99	0.95	0.12--	0.64
miscellan	1.02	1.11		0.79	0.68	0.95
optimizat	0.99	1.35	0.78	1.01	0.90	0.86
program_o	0.98	0.93	0.86	1.41	1.07	1.05
statement	1.04	1.18	0.66-	0.91	1.06	0.93
storage_r	0.99	0.99	1.41	1.25	0.96	0.97
subprogra	1.08	0.94	0.59--	0.94	1.21	0.84
systemati	1.10	0.88		0.78	0.79	1.02
tasking	1.50	0.81	1.72++	0.83	0.85	0.95
Sys Fact	1.08	1.24	0.31	0.58	1.59	

Execution Times --- 20 Dec 1991 12:14:16 --- Page 2

Summary over all groups : product data - product model

---- System Names and Descriptions

```
v_91
  -- Host: VAXstation 3100                 Target: VAXstation 3100
t_91
  -- Host: VAXstation 3100                 Target: VAXstation 3100
s_91
  -- Host: MIPS R2000A/R3000               Target: MIPS R2000A/R3000
m_91
  -- Host: DECstation 3100 MIPS RISC       Target: DECstation 3100 MIPS RISC
d_91
  -- Host: VAX 6220                        Target: 1750A
```

---- System Factors and Confidence Intervals (including graph)

Systems	Low	Mean	High	Ratio	0.2 ... 1.8
v_91	0.94	1.08	1.25	1.00	|--+---|
t_91	1.09	1.24	1.40	1.14	|--+---|
s_91	0.23	0.31	0.43	0.29	-+--|
m_91	0.46	0.58	0.74	0.54	|--+---|
d_91	1.39	1.59	1.82	1.47	|----+----|

---- Significant Diff = *

---- Data Summary: Total n = 1627

	v_9	t_9	s_9	m_9	d_9	Gps	Valid	NoData	Comp	RunTim	Exclu	Other
v_91		-	*	*	*	17	1541	0	14	1	25	46
t_91	-		*	*	-	17	1496	29	31	12	1	58
s_91	*	*		*	*	15	894	30	82	15	2	604
m_91	*	*	*		*	17	1444	81	42	21	0	39
d_91	*	-	*	*		17	1175	319	71	22	1	39

---- Group Weights

application	1.0	arithmetic	1.0	classical	1.0
data_storage	1.0	data_structures	1.0	delays_and_timing	1.0
exception_handlin	1.0	generics	1.0	input_output	1.0
miscellaneous	1.0	optimizations	1.0	program_organizat	1.0
statements	1.0	storage_reclamati	1.0	subprograms	1.0
systematic_compil	1.0	tasking	1.0		

Execution Times --- 20 Dec 1991 12:14:16 --- Page 1

AES Review Outline

Background
Test Harness
Specific Test Problems
Assessors
Analysis / Reporting

AES 2.0 REVIEW BACKGROUND

Read documentation and code for performance tests
Executed the performance test groups
Reviewed documentation and ran other selected groups
AES design philosophy assumes testing service
- Expected to develop core of people experienced with porting
- Relatively small number of performance tests
  - No automated system comparison tool
- Emphasizes textual reporting
  - Qualitative findings (optimization performed or not)

AES provides broad coverage of capabilities of "whole APSE"

TEST HARNESS

- Does not delete unneeded program library units
- Harness error messages not helpful
- New database needed to evaluate tools
- Manual mode doesn't clearly explain "work/not work"
- Doesn't document where reports are placed
- RESULTS.DBS was easily corrupted

TEST HARNESS PORTING EFFORT

- AES forces users to learn operating system interfaces for
  - Control of split screen
  - Spawning processes
  - Invoking job control statements from Ada program
- Requires information about AES internal structures
- Requires adaptation of preprocessor
- Desires compiler supporting all Ada with OS interfaces

OVERVIEW OF FLAWED PERFORMANCE TESTS

14 of 19 tests in Group I (General Runtime Efficiency) are flawed
9 of 20 tests in Group O (Optimization) are flawed

Flawed tests can be corrected but all require review

Representative examples are presented

CRITERIA FOR PERFORMANCE TESTS

With respect to the timing loop, each test problem should:

Not be loop invariant
Use live variables
Contain expressions that are not strength reducible
Not be unduly foldable
Follow same path on each repetition
Use initialized variables

AES EXAMPLE 1: TI01D

Test problem contains 40 statements of the form
Intended to test efficiency of record comparision

```
SAME := A1 = A2 ;   -- where A* is a record
SAME := A2 = A3 ;
...
NOT_SAME := (A1 /= A2) and (A3 /= A4);
NOT_SAME := (A5 /= A6) and (A7 /= A8);
...
```


FLAWS IN EXAMPLE 1

38 of 40 assignments are dead

ALL expressions are loop invariant

Even if record comparisons could raise exceptions, LRM would explicitly permit reordering (11.6)

AES EXAMPLE 2: TO02

Examples RL11 & RL12 are intended to detect whether common subexpressions involving two-dimensional array addressing for integers are recognized as common

```
RL11 =>  RAA  (K, L) := RBB  (K, L) + RV;
         RAA2 (K, L) := RBB2 (K, L) + RV;

RL12 =>  RAA  (K, L) := RAA  (K, L) + RV;
         RAA2 (K, L) := RAA2 (K, L) + RV;
```

AES assumes RL12 will be smaller than RL11 iff common subexpressions are recognized

FLAWS IN EXAMPLE 2

- Array TYPES are identical so the ALL subscripting expressions are common
- All subscripting expressions are loop invariant
- Use of ADD_TO_MEMORY instruction can confound interpretation
- No "credit" for recognizing that expression could be evaluated once
- Conclusions drawn from test results can be wrong

AES EXAMPLE 3: TI11

The third part of this problem calculates the FOR loop overhead

```
Version 1 => if not TRUE1 then  -- done twice as often as version_2
                K := K + I2; A (K) := K;  -- I2 is timing loop index
             end if;
Version 2 => if not TRUE1 then
                K := K + I2; A (K) := K;
             end if;
             if not TRUE2 then
                L := L + I2; A (L) := L;
             end if;
```

Reports "overhead" as (time_for_v1 - time_for_v2) / iterations_of_v1

FLAWS IN EXAMPLE 3

- TRUE1 & TRUE2 are loop invariant
- Instruction prefetching will favor second example
- The idea that a system has a constant FOR LOOP overhead is flawed
  - An optimizer may unroll some loops, reducing loop overhead
  - Complexity of body may permit/prevent keeping FOR index in register
  - Size determines whether code can use long/short format instructions
  - Memory effects: cache and/or prefetching and/or "loop mode"
  - Target processor may have idiomatic instructions
  - After loop invariant motion, body might be null
  - Strength reduction on FOR loop index used only as subscript

AES EXAMPLE 4: TR23

Test problem to detect whether compiler does loop motion:

```
for I in ONE_TO_TEN loop
   SUM (I) := S + A (I);
   A (INDEX) := S;      -- INDEX is out-of-range
end loop;
```

Tests whether value of SUM(1) has been modified

FLAWS IN EXAMPLE 4

"A(INDEX) := S;" only invariant if flow analysis determines INDEX is never one

Confounds loop invariant motion with data flow analysis

AES TIMING LOOP

- Does not subtract off null loop time (documentation says it does)
- Does not distinguish between inconsistent measurements and test failures
- Uses fixed number of outer loop cycles
- Bases inner loop count on clock tick
- Does not use confidence levels
- Uses nested FOR loops

ASSESSORS

Includes assessors for non-Ada specific capabilities

Requirements analyzer	Version Configuration Control
Editors (general purpose)	Command Language Interpreter

Includes assessors for various Ada-specific capabilities

Compiler/Linker diagnostics	Compiler/Linker capacity limits
Compiler performance	Runtime performance
Name expander	Pretty printer
Source generator	Timing analysis tools
Test coverage tool	Stub generator
Assertion checker	Cross reference analyzer
Syntax oriented editor	Testbed generator

ANALYSIS / REPORTING

AES does not provide comparative analysis tool
AES reports states conclusions without supporting data
  TO02 reports whether optimization performed or not
AES presents a lot of descriptive text along with results

SUMMARY

Test Harness
  Inappropriate to non-test service based usage
Test Problems
  Many contain flaws which should be corrected
Assessors
  AES provides broad coverage of APSE capabilities
Analysis
  Lack of automated system comparisons is significant

Carnegie Mellon University
Software Engineering Institute

Comments on the Ada Evaluation System

January 22, 1992 (Version 1.00)

Software Engineering Institute
Carnegie Mellon University
Pittsburgh PA 15213

Sponsored by the U.S. Department of Defense

1

Carnegie Mellon University
Software Engineering Institute

Background

The SEI has had access to the AES for about four years, and has used it in several ways:

- **As software subjected to critical review**
- **As an element in a benchmark tutorial**
- **As an evaluation tool in performance analysis**

The AES elements considered in this talk:

- **Executable benchmark tests (primary)**
- **Test harness (primary)**
- **Check lists (secondary)**
- **Documentation (secondary)**

2

Organizational Issues

Consistent use of Test Categories

- **Tests are organized into groups**
  1. **Object under test (compiler, loader, etc.)**
  2. **Concept under tests (optimizations, etc.)**
- **Categories are used for**
  - **Documentation**
  - **File Naming**
  - **Testing**
  - **Reports**

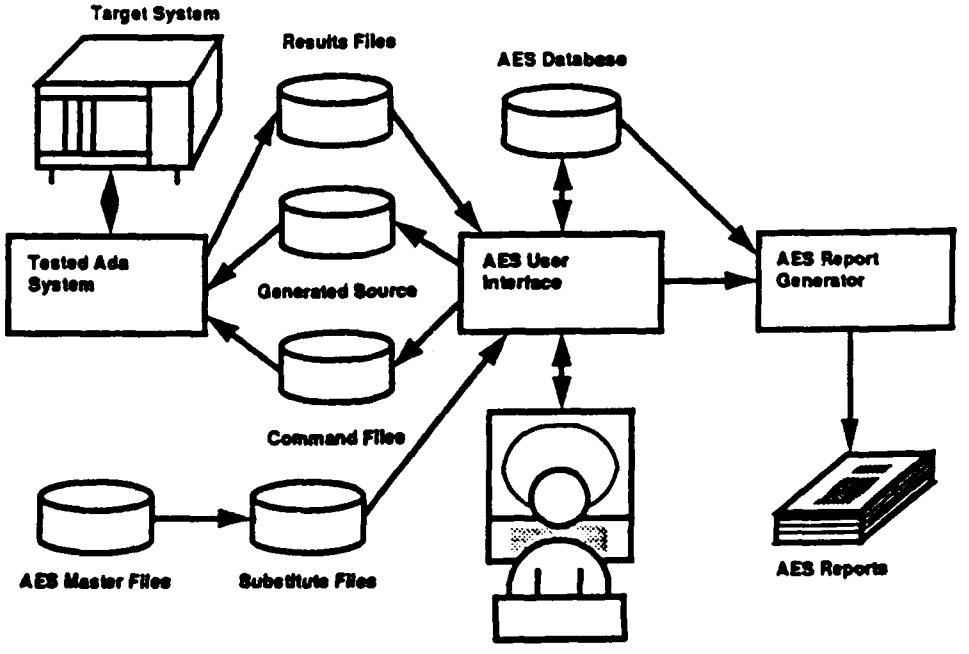
Convenient Access to Data (Database)

- **Data stored as text**
- **Database used for results and control variables**
- **Only one value per database item (e.g. one compiler per database, one test run only)**

3

Carnegie Mellon University
Software Engineering Institute

File Organization



4

User Interface -1

Strengths

- **Interactive**
- **Integrates all important components**
  - **Set up**
  - **Database**
  - **Testing**
  - **Report generation**
- **Automatic or manual mode using same model**
- **Script driven, flexible**

User Interface -2

Weaknesses

- **Modal (set up mode, test mode, report mode)**
- **Undocumented commands**
- **Partial memory of previous state**
- **Portability complicated by screen interface**
- **Weak help and tutorial**
- **Database access is limited to raw data**
- **No provision for subsetting or exchanging data**

Scripts

Scripts provide

- **Conditional testing based on test results**
- **For user modifications**
- **Convenient substitution points for system dependencies**

Script problems

- **Hard to identify truly universal operations (e.g. Verdix does listing with a separate utility, not through compiler)**
- **Sometimes no handle provided (e.g. Verdix uses ".a" for all source files)**
- **Operations may be divided or combined (e.g. optimization: pragma, command line switch or both)**

Focused Testing

Some of my favorite tests/methods:

- **Limit testing (using binary chop)**
- **Memory allocation strategies (three models)**
- **Variation in implementation (e.g. case implementation strategies)**

Test Issues

Identified test needs

- Measuring individual features
- Compare different systems
- Compare different versions
- Time statements
- Coding for performance

AES vs. ACEC

- Measuring individual features: AES better
- Compare different systems: AES worse
- Compare different versions: AES equal
- Time statements: AES much worse
- Coding for performance: AES better

Timing

Features

- Ignores certain biases
- Selects timing based on clock resolution
- Portable, only needs standard clock
- Tests for consistency of results
- Time value is average of all tests, not minimum

Issues

- No provision for automated substitution
- Not suited for fast timers
- Can only generate average values
- Timing can start before I/O concludes

Bias Control

AES controls for certain kinds of bias:

- Memory location effects: averaged values by multi-statement segments
- Timing overhead: tested code segments kept large, timing overhead is assumed to be small
- Timing variation: multiple runs, tests which show significant variation are marked as failing

11

Missing but Desirable

- Open systems: features and documentation to support customization and extension
- Presentation: flexible report and graphic output
  -As part of standard presentations
  -To allow interactive testing
- Data manipulation: need to select and exchange data
  -For spreadsheets
  -Raw data for statistical packages
  -In table format for text processing

12

Carnegie Mellon University
Software Engineering Institute

Comments on the Preliminary Release of ACEC 3.0

22 January 1991

Software Engineering Institute
Carnegie Mellon University
Pittsburgh PA 15213

Sponsored by the U.S. Department of Defense

Carnegie Mellon University
Software Engineering Institute

Background

SEI has been involved with ACEC for the past few years

Release 2.0 has been used in the past year
- **As software subjected to critical review**
- **As an element in a benchmark tutorial**
- **As an evaluation tool in performance analysis**

Preliminary release 3.0 was received in mid-December, 1991, so an extensive evaluation of it has not been performed

Carnegie Mellon University
Software Engineering Institute

SEI Work to Date with Release 3.0

All documents read and all pre-test steps run

Limited number of performance tests run

Problems encountered running SSA and Menu

CA and assessor tools not run

SEI Configuration

- **Compiler: Verdix VADS VMS - MC68030 6.0.5(f)**
- **Host: DEC MicroVAX 3200 running VMS 5.3**
- **Target: 25 MHz Motorola MC68030**
- **Compiler for host-specific analysis tools was DEC VAX Ada 2.1**

Carnegie Mellon University
Software Engineering Institute

Overall Comments

The User's Guide and the organization of the software are much improved over the previous release

- **Logical step-by-step guide to pre-tests**
- **Pre-tests incorporate actual execution of tests and analysis tools**
- **Command file naming conventions and division of tests into performance groups makes life easier**

The Reader's Guide still needs work to achieve the clarity and usefulness of the User's Guide, particularly the sections dealing with the Comparative Analysis tool

The pre-test and test suite command files do not adequately address the needs of host-target systems

Carnegie Mellon University
Software Engineering Institute

Comments on the Reader's Guide

The Reader's Guide doesn't yet clearly answer the question: How does the ACEC compare compiler C1 on machine M1 with compiler C2 on machine M2?

There should be a clear statement of the level of knowledge required of a user for correct interpretation of all analysis tool outputs.

The sections on the Comparative Analysis tool's output and background need to be re-organized and made more understandable.

The chapter on timing techniques is good but needs some re-organization to make it more user-friendly

Carnegie Mellon University
Software Engineering Institute

Some Problems Encountered

TCAL1 and TCAL2 tests (pre-test step 4) didn't work

Double-precision math library test of Power function failed with an Argument_Error exception

SSA tool couldn't open database file created by the Condense tool; Menu program subsequently crashed

Menu program crashed immediately when "PS:" was specified in a VMS pathname in the System Names file

Concluding Remarks

Provide users with some estimate of how long it takes to set up and run the ACEC and analyze the results

Emphasize the need for users to treat running ACEC as part of a larger overall evaluation PLAN

Think about graphical output and/or output suitable for analysis by spreadsheets

Think in terms of benchmark generation rather than benchmark instantiation

Hypertext Version of the E&V Reference System: A Sampler

23 January 1992

[Sample material from the User's Guide and system screens]

Prepared by:

Bard S. Crawford
TASC
55 Walkers Brook Drive
Reading, MA
01867

617-942-2000
crawford@ajpo.sei.cmu.edu

1. INTRODUCTION

The Ada Programming Support Environment (APSE) Evaluation and Validation (E&V) Reference System is a pair of documents developed, and periodically updated, by the APSE E&V Project, sponsored by the Ada Joint Program Office and led by the US Air Force Avionics Directorate of the Wright Laboratory. The documents are entitled the "E&V Reference Manual" and the "E&V Guidebook."

The **E&V Reference Manual** provides a framework for understanding APSEs and their assessment, and establishes common terminology. One chapter discusses an APSE as a whole and its assessment. Other chapters are indexes to APSE component characterization and assessment, organized by life cycle activities, APSE tool category, APSE function, and attribute to be assessed. An entry in an index consists of a description, cross references to other entries in the Reference Manual, and cross references to the "E&V Guidebook." The manual is intended to help a variety of users obtain answers to their questions. As a stand-alone document it is intended to help a user find useful information about index elements and relationships among them. In conjunction with the Guidebook, it is intended to help users find criteria and metrics for assessment of APSEs and their components.

The **E&V Guidebook** provides descriptions of specific instances of assessment technology. These include evaluation (assessment of performance and quality) and validation (assessment of conformance to a standard) techniques. For each category of item to be assessed (e.g. compilation system, test system, whole APSE, etc.), there are brief descriptions of applicable tools and aids -- such as test suites, questionnaires, checklists, and structured experiments -- and references to primary documents containing detailed descriptions. The Guidebook also contains synopses of documents of general historical importance to the entire field of Ada environments and their assessment.

Hard copy versions (1.1, 2.0, 3.0) of both documents have been published beginning in 1987. These are available through the Defense Technical Information Center (DTIC). The Version 3.0 DTIC numbers are:

AD A236 697 – E&V Reference Manual
AD A235 494 -- E&V Guidebook

The text of the hypertext version (3.1) is based on the most recent hard copy version (3.0), published in February 1991 -- with a few minor additions and corrections. The hypertext version runs on Macintosh® computers as a set of Hypercard® stacks. It requires Hypercard Version 2.0 or later. It is shipped as a set of three 3.5 inch double-density, double-sided disks, plus this document.

Disk A contains 8 stacks -- two special stacks and part of the E&V Reference Manual.
Disk B contains 3 stacks -- Chapters 6 and 7, and Appendices of the E&V Reference Manual.
Disk C contains 18 stacks -- the entire E&V Guidebook.

E&V-Maps and **E&V-Help** are the two special stacks mentioned above. The other 27 stacks correspond directly to textual material previously published in the most recent hard copy version (3.0) -- except for the many "hyperlinks" and navigation devices incorporated along with the text in the hypertext version (3.1).

2. INSTALLATION AND START-UP

You must have a Macintosh with Hypercard Version 2.0 or later already installed. The E&V Reference System stacks require approximately 1.9 megabytes of memory on your hard disk.

It is very easy to install the system for anyone familiar with the Macintosh desktop system for creating, copying, and dragging folders and files. Perform the following steps to install the system:

1. Create a new folder with a name such as "E&V RefSys"
2. Copy the contents of disk A (8 files) into your new folder.
3. Copy the contents of disk B (3 files) into your new folder.
4. Copy the contents of disk C (18 files) into your new folder.
5. Arrange the 29 icons representing the 29 files in a neatly organized manner such as that shown in Fig. 2-1.

To start the system running, you can double-click on any of the 29 icons. Normally, you will want to start from a high-level view. The way to do this is to double-click on the **E&V-Maps** icon. This takes you to the first card in the stack; it is called **Top-Level E&V Map.** If you are already familiar with the system based on past experience, you may want to go directly to one of the chapters by double-clicking on the icon corresponding to that chapter. You can easily get to the help screens from every card in any of the other 28 stacks, by clicking on the ? button. You can also start in the **E&V-Help** stack by double-clicking on the icon with that name. In fact, a good way to begin, if this is your first look at the system, is to go directly to the **E&V-Help** stack and browse through the first section called **Welcome and Introduction.** The help screens are also printed out in Section 3 of this User's Guide.

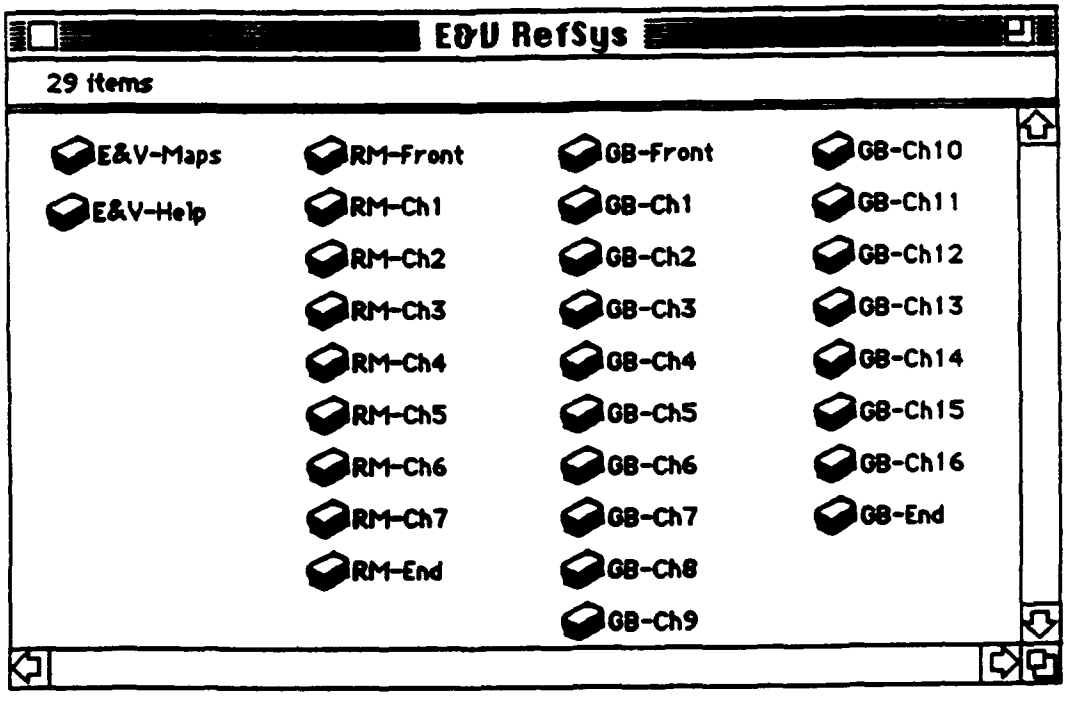
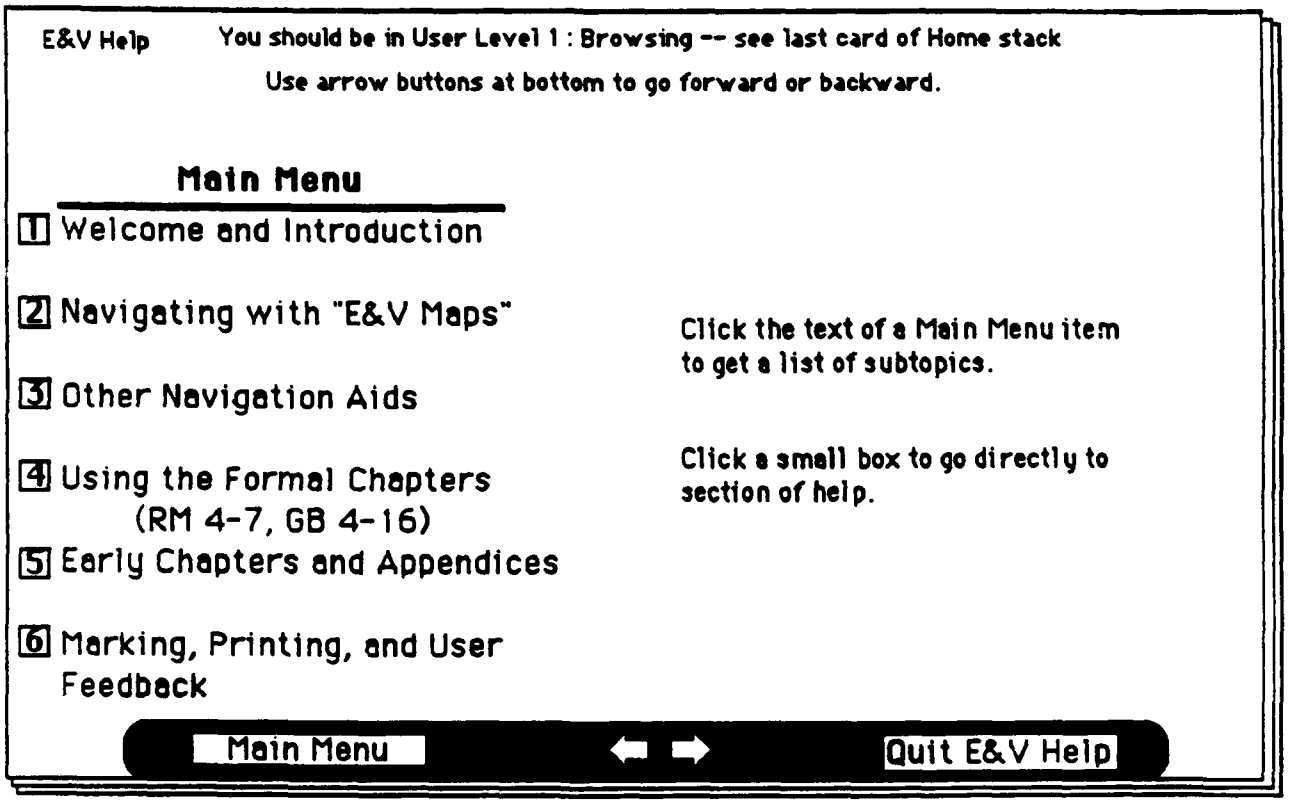
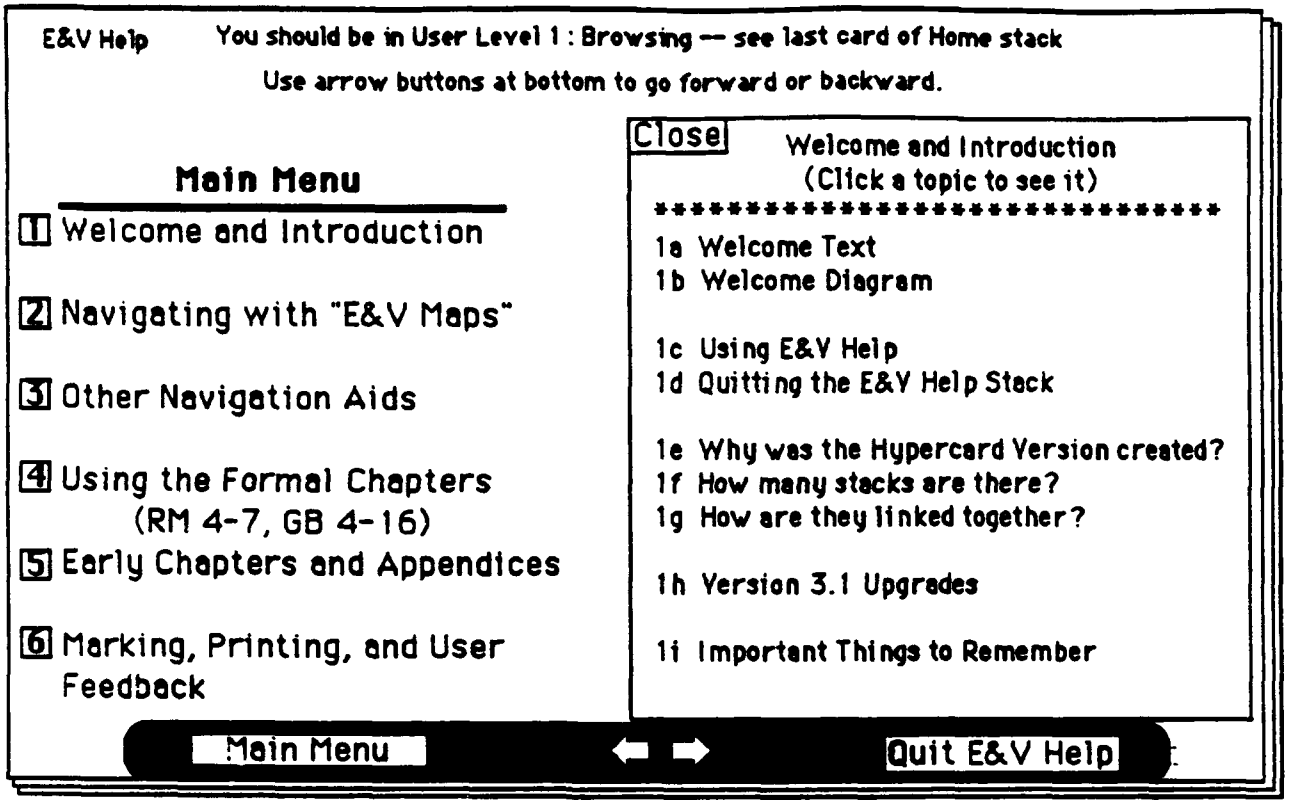


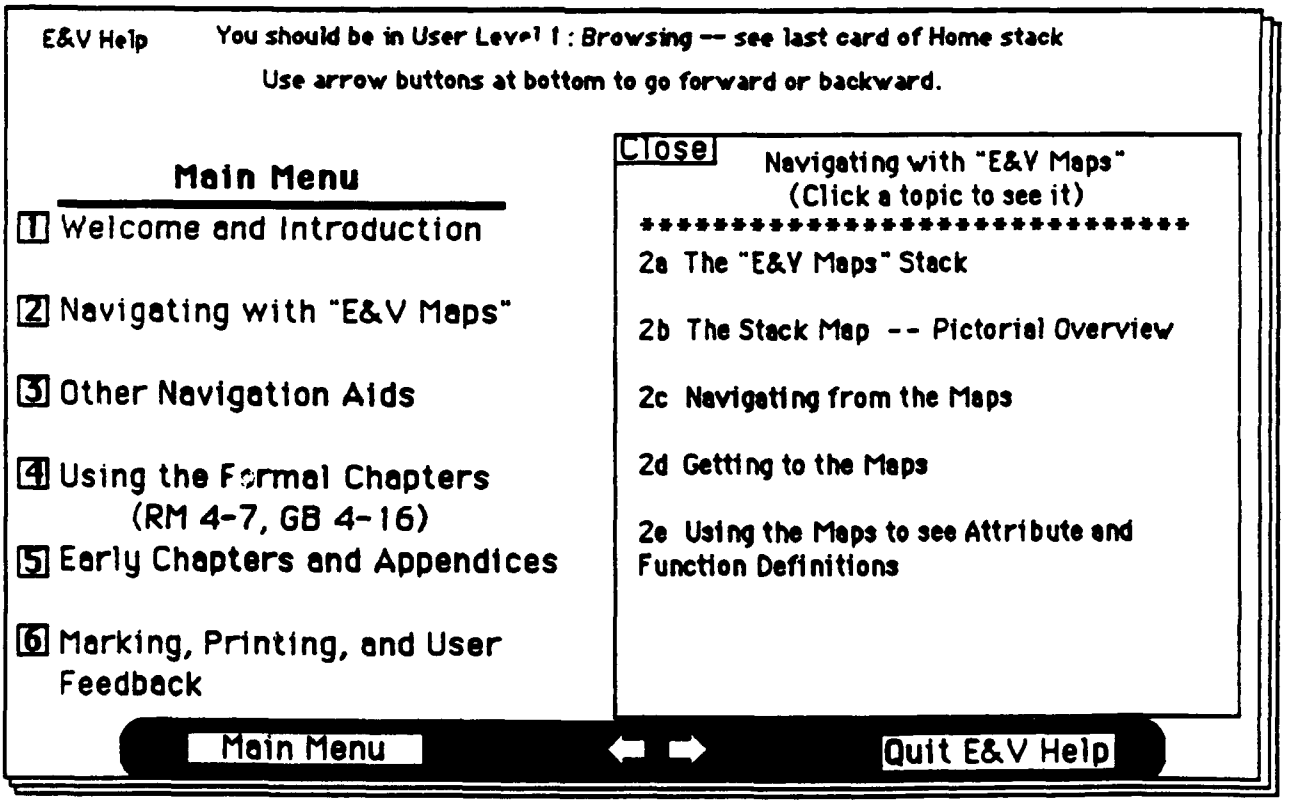
Figure 2-1 Suggested Arrangement of Stack Icons

3. HELP SCREENS: PRINT-OUT

This section provides a print-out of the complete set of help screens in the **E&V-Help** stack. The first card in the stack is the help **Main Menu.** The other cards are grouped into six subdivisions, as indicated in the Main Menu. The Main Menu is printed out seven times in the next four pages, each time with a different choice of "pop-up fields" displayed. This provides, in effect, a Table of Contents for the complete set of help screens. The remainder of the cards provide an overview of the contents of the entire system and how to use it.




E&V Help
You should be in User Level 1 : Browsing -- see last card of Home stack
Use arrow buttons at bottom to go forward or backward.
Main Menu
1 Welcome and Introduction
2 Navigating with "E&V Maps"
3 Other Navigation Aids
4 Using the Formal Chapters (RM 4-7, GB 4-16)
5 Early Chapters and Appendices
6 Marking, Printing, and User Feedback
Close
Welcome and Introduction
(Click a topic to see it)
1a Welcome Text
1b Welcome Diagram
1c Using E&V Help
1d Quitting the E&V Help Stack
1e Why was the Hypercard Version created?
1f How many stacks are there?
1g How are they linked together?
1h Version 3.1 Upgrades
1i Important Things to Remember
Main Menu
Quit E&V Help


E&V Help
You should be in User Level 1 : Browsing -- see last card of Home stack
Use arrow buttons at bottom to go forward or backward.
Main Menu
1 Welcome and Introduction
2 Navigating with "E&V Maps"
3 Other Navigation Aids
4 Using the Formal Chapters (RM 4-7, GB 4-16)
5 Early Chapters and Appendices
6 Marking, Printing, and User Feedback
Close
Navigating with "E&V Maps"
(Click a topic to see it)
2a The "E&V Maps" Stack
2b The Stack Map -- Pictorial Overview
2c Navigating from the Maps
2d Getting to the Maps
2e Using the Maps to see Attribute and Function Definitions
Main Menu
Quit E&V Help

E&V Help — Welcome and Introduction

1a Welcome Text

(Note: be sure to click the "balloon" and then click the "close box")

Welcome to the Hypercard version of the **E&V Reference System** -- Version 3.1.

The system helps users make assessments of tools, tool sets, and environments (**APSEs**). It contains two electronic "hyperdocuments:"

"E&V Reference Manual"
"E&V Guidebook."

Assessments fall into two categories:

Evaluation (E) is assessment of performance and quality.
Validation (V) is assessment of conformance to a standard.

Main Menu ⇦ ⇨ Quit E&V Help

E&V Help — Welcome and Introduction

1a Welcome Text - continued

Hard copy versions (1.1, 2.0, and 3.0) have been published beginning in 1987. These are available through the Defense Technical Information Center (DTIC). The Version 3.0 DTIC Numbers are:

E&V Reference Manual -- AD A236 697
E&V Guidebook -- AD A236 494

This electronic version (3.1) is based on the most recent hard copy version (3.0), published in February 1991. Most of the text is the same -- the exceptions to this rule are indicated in "Version 3.1 Upgrades" (see E&V Help Item 1h). The early chapters of both documents contain background material on the need for E&V and the history of the E&V Project.

Main Menu ⇦ ⇨ Quit E&V Help

E&V Help

Welcome and Introduction

1a Welcome Text - continued

The next card is a pictorial representation of the two documents and their relationship to one another.

Other graphical representations of the system may be found in the stack called **E&V Maps** -- a very important stack, which you should be sure to read about a little later in this **E&V Help** stack.

E&V Help and **E&V Maps** are of course not included in the hard copy version of the system.

Main Menu ⇦ ⇨ Quit E&V Help

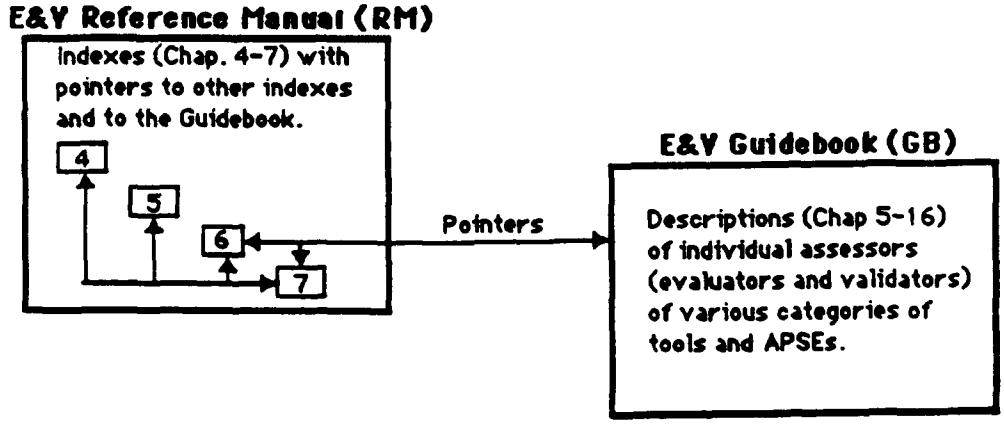
E&V Help

Welcome and Introduction

1b Welcome Diagram

Welcome to the Hypercard version of the "E&V Reference System."

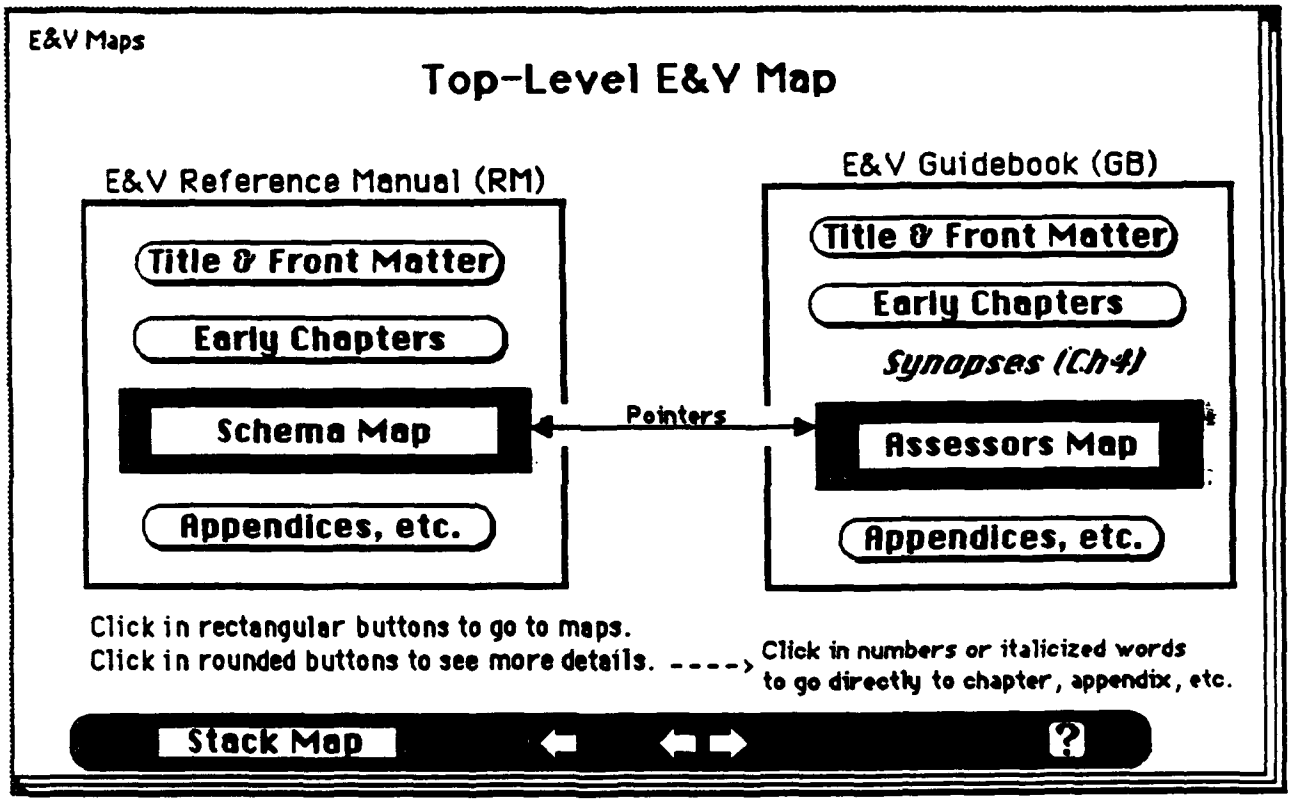
The system consists of two documents, which contain many inter-document and intra-document links (pointers), as indicated pictorially below.



Main Menu ⇦ ⇨ Quit E&V Help

4. E&V MAPS: PRINT-OUT

This section provides a print-out of the eight cards of a special stack called **E&V-Maps**. The on-screen versions of most of these cards provide a great deal of hidden information -- available in pop-up fields. You will not, of course, have access to the hidden information in this printed form. But, you can see how the top-level access is organized, and you can read about the mechanisms involved by reviewing Part 2 of the **E&V-Help** stack given previously.



E&V Maps

Functions Map

Guide to Chapter 7. **Functions** of the E&V Reference Manual

7.1 Transformation

- 7.1.1 Editing
- 7.1.2 Formatting
- 7.1.3 On-Line Assistance
- 7.1.4 Sort/Merge
- 7.1.5 Graphics Generation
- 7.1.6 Translation
- 7.1.7 Synthesis

7.2 Management

- 7.2.1 Information Management
- 7.2.2 Project Management
- 7.2.3 Computer System Mgmt

7.3 Analysis

- 7.3.1 Static Analysis
- 7.3.2 Dynamic Analysis
- 7.3.3 Formal Verification
- 7.3.4 Problem Report Analysis
- 7.3.5 Change Request Analysis

Click number to go to section.
Click function name to see description.
Click round button to see lower-level details.

Top-Level E&V Map

E&V Maps

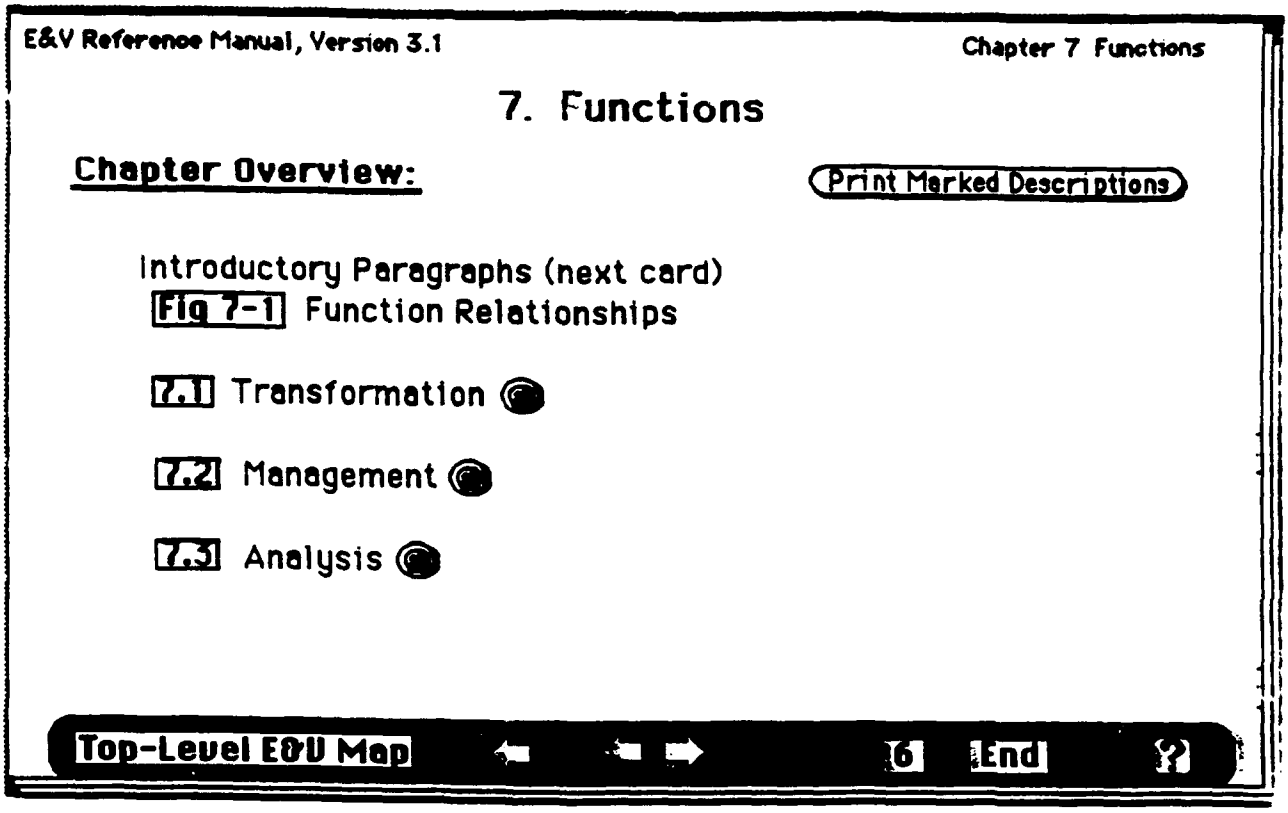
Assessors Map

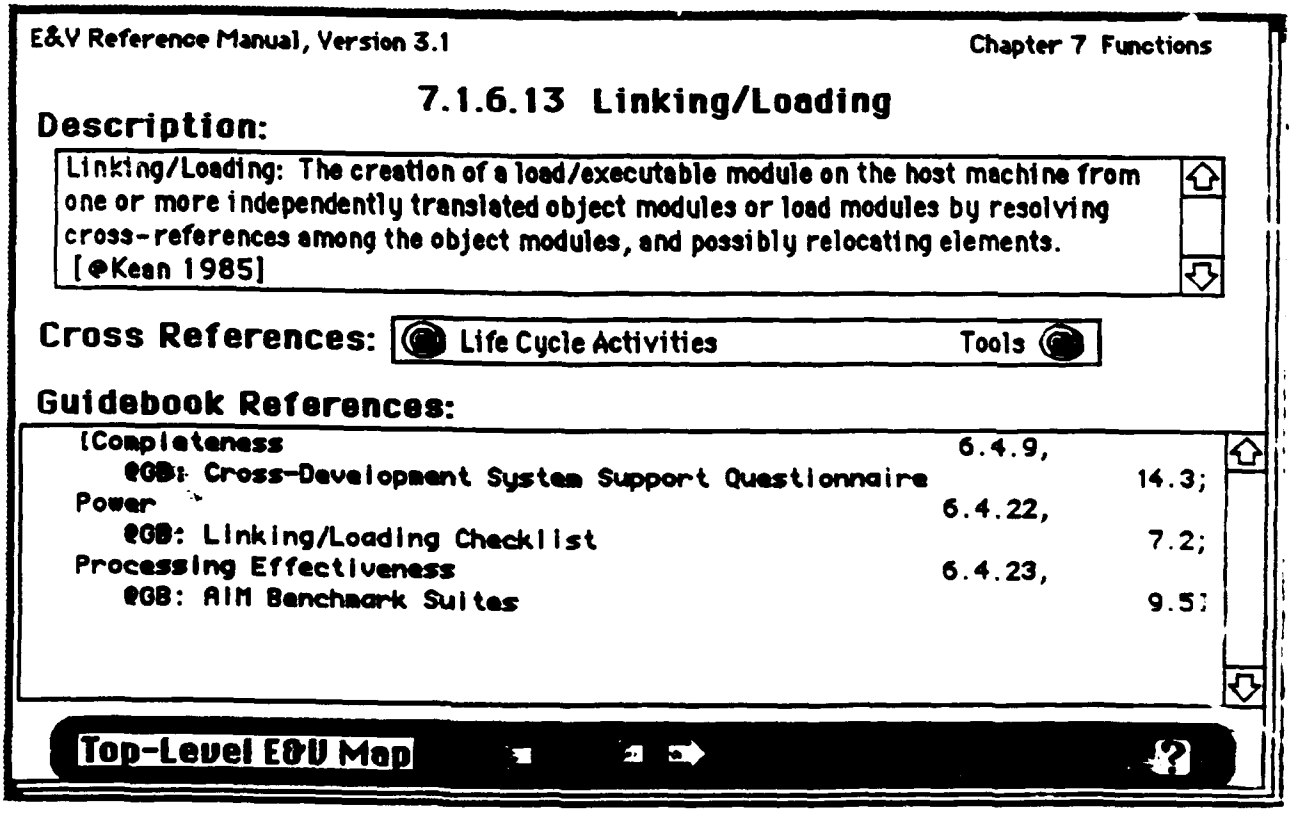
Guide to Chapters 5-16 of the E&V Guidebook

5. General Purpose Assessors
6. Compilation System Assessors
7. Target Code Generation Aids and Analysis Toolset Assessors
8. Test System Assessors
9. Tool Support Component Assessors
10. Requirements/Design Support Assessors
11. Configuration Management Support Assessors
12. Distributed System Dev'mt and RTS Assessors
13. Distributed APSE Assessors
14. Whole APSE Assessoprs
15. Information Management Support Assessors
16. Other Assessors

Click number to go to section.
Click round button to see lower-level details.

Top-Level E&V Map


E&V Reference Manual, Version 3.1
Chapter 7 Functions
7. Functions
Chapter Overview:
Print Marked Descriptions
Introductory Paragraphs (next card)
Fig 7-1 Function Relationships
7.1 Transformation
7.2 Management
7.3 Analysis
Top-Level E&V Map
6
End


E&V Reference Manual, Version 3.1
Chapter 7 Functions
7.1.6.13 Linking/Loading
Description:
Linking/Loading: The creation of a load/executable module on the host machine from one or more independently translated object modules or load modules by resolving cross-references among the object modules, and possibly relocating elements. [@Kean 1985]
Cross References:
Life Cycle Activities
Tools
Guidebook References:
{Completeness 6.4.9,
@GB: Cross-Development System Support Questionnaire 14.3;
Power 6.4.22,
@GB: Linking/Loading Checklist 7.2;
Processing Effectiveness 6.4.23,
@GB: AIM Benchmark Suites 9.5;
Top-Level E&V Map

E&V Reference Manual, Version 3.1 Chapter 7 Functions

7.1.6.7 Compilation

Description:

Compilation: Translating a computer program expressed in a procedural or problem-oriented language into object code. [ΦKean 1985]

Cross References: Life Cycle Activities Tools

Guidebook References:

ΦGB: Ada Compiler Specification and Selection Questionnaires 6.18);
Processing Effectiveness 6.4.23,
(*ΦGB: IDA Benchmarks 6.2,
*ΦGB: Ada Compiler Evaluation Capability (ACEC) 6.3,
*ΦGB: PIWG Benchmark Tests 6.4,
*ΦGB: University of Michigan Benchmark Tests 6.5,
*ΦGB: UK Ada Evaluation System (AES) 6.7,
*ΦGB: ARTEWG Catalogue of Ada Runtime Implem. Dependencies 6.10,
ΦGB: Compiler Assessment Questionnaire 6.12.

Top-Level E&V Map

E&V Guidebook, Version 3.1 Chapter 6. Compilation System Assessors

6.3 Ada Compiler Evaluation Capability (ACEC)

Purpose | Primary References | Vendors/Agents | Method

6.3 Ada COMPILER EVALUATION CAPABILITY (ACEC)

Purpose: The Ada Compiler Evaluation Capability (ACEC) Version 2.0 was developed by Boeing Military Airplanes for the Ada Joint Program Office (AJPO) under the direction of the Air Force Wright Research and Development Center (WRDC). Its primary purpose is to provide the capability to determine the performance and usability characteristics of Ada compilation systems. The ACEC consists of the ACEC Software Product and three supporting documents: the ACEC User's Guide, the ACEC Reader's Guide, and the ACEC Version Description Document.

ACEC Software Product - The ACEC Software Product consists of performance tests, assessor tools, and support software. The software product makes it possible to:

- Compare the performance of several implementations

Top-Level E&V Map

6. Compilation System Assessors

Chapter Overview

Print Marked Assessors

Introductory Paragraphs (next card)

Top-Level E&V Map

6.7 UK Ada Evaluation System (AES)

Purpose Primary References Vendors/Agents Method

6.7 UK Ada EVALUATION SYSTEM (AES)

Purpose: Evaluation of Ada compilers and associated linkers/loaders, program library systems, debuggers, and run-time libraries. A test suite and a methodology (AES) were developed by Software Sciences Ltd., under sponsorship of the UK Ministry of Defense (MoD). The British Standards Institute (BSI) has been sponsored by the MoD to provide an Ada Evaluation Service, using the AES. Interested parties, such as compiler vendors or potential compiler purchasers, may pay BSI to conduct an evaluation or to supply a copy of an existing evaluation report.

[@RM: Compilation	7.1.6.7,	(@RM: Accuracy	6.4.1,
		@RM: Anomaly Management	6.4.2,
		@RM: Capacity	6.4.6,
		@RM: Cost	6.4.11,
		@RM: Operability	6.4.21,
		@RM: Processing Effectiveness	6.4.23,

Top-Level E&V Map

E&V Guidebook, Version 3.1 Chapter 8. Test Systems Assessors

Table 8.1-1 Testing Capabilities Checklist

Table 8.1-1 Testing Capabilities Checklist

FEATURE	FOUND	NOTES
Static Analyzers		
Code Auditors		
Consistency Checkers		
Interface Analyzers		
Completeness Checkers		
Tool Building Services		
Common ``Front-End'' Facilities for Languages of Interest (Parsing, Source & Internal Form Manipulation, Execution Facilities)		
Tool Composition Aids		

Top-Level E&V Map

E&V Guidebook, Version 3.1 Chapter 13. Distributed APSE Assessors

Figure 13.1-1 Distributed APSE Questionnaire

Architecture
- Type of Distribution
  - What is distributed on the APSE: processing resources, data, or both?
- Heterogenous/Homogenous
  - Does the APSE support a heterogenous hardware configuration or is it restricted to implementation on a homogenous hardware configuration?
    - Is there special hardware required for its implementation on a heterogenous configuration?
    - Are there special software communication protocols that are required for implementation on a heterogenous configuration?
- Node Transparency
  - Is the same toolset available on all nodes in the APSE?
    - If so, how is the commonality defined (e.g., common user interface, common functionality, and support by a common vendor)?
    - If not:
      - Is the user-interface and functionality the same across all nodes?

Top-Level E&V Map

ACEC - AES Merger

Objectives
Interface
Performance Tests
Analysis
Assessors

Objectives

Portability
Ease of adaptation
Ease of use
Minimize cost/benefit ratio for users
Upward compatability (200 ACEC users)

Take the best of the AES and add it to the ACEC

Resource Constraints

Interface Issues: The Test Harness

Database

Preprocessor
- Command file generation
- Ada program generation

Direct execution mode

Database

Keeps track of progress

Insure that checkout tests have been run

Using information from checkout tests

Command File Generation

- Ease the adaptation process
- Do not have to remember file names
- Tailoring
  - Database information
  - User requests

Ada Program Generation

- Conserve disk space
- Parameterization
- System Adaptation

Direct Execution Mode

- Adaptation difficulty
- Some tests are NOT automatable
  - Debugger
  - Diagnostics
- Interactive versus batch
  - Number of tests
  - Running time

Adaptation

- Command language
- Tool specific commands
- Ada/Operating system interface
- Screen control

The ACEC under the Test Harness

Pretest	Interactive / Manual
Entering results	Automatic / Manual
Performance tests	
Groups	Batch
Individual tests	Interactive / Direct
Entering results	Automatic
Analysis	Interactive / Batch

The ACEC under the Test Harness

Assessors	
Debugger	Interactive / Manual
Entering results	Manual
Diagnostics	Interactive / Manual
Entering results	Manual
Library	Interactive / Manual / Batch
Entering results	Manual / Automatic
Capacity	Batch
Entering results	Automatic

AES Performance Tests

Review and integrate into ACEC groups

Use ACEC timing loop

Provide for automatic gathering of results

Integrate in Comparative and Single System Analysis tools

Performance Tests: AES / ACEC Map

AES group		ACEC group
A	compiler performance tests	SY
I	general run-time efficiency tests	various
J	NPL Performance Test Suite	various
K	tasking tests for MASCOT systems	TK
L	general tasking tests	TK
M	storage management tests	SR
N	input/output tests	IO
O	optimization tests	OP
R	implementation dependency tests	various
V	benchmark tests	CL

Analysis

Menu

Input data

Single System Analysis - AES reports

Comparative Anlysis

AES Assessors

Similar Assessors

Candidates for inclusion

Non-candidates

Asessors: AES / ACEC Map

AES group		ACEC group
B . . F	compiler information tests	YD
G	compiler capacity tests	YC
Q	run-time limit tests	YC
S	erroneous execution tests	YD
T	incorrect order dependency tests	YD
U	linker/loader tests	YL,SY
D*	debugger tests	YB
LS	PLS scenario support tests	YL
S*	source generator tests	YL

Debugger Assessor: AES versus ACEC

	AES	ACEC
Number of questions	272	118
Test programs	11 +	36
Scenarios	5	29
Detail of scenarios	less	more

Debugger Assessor: AES

Harness
- Generates command files, capacity tests
- Prompts for some results
- Generates report for capacity

Questionnaires

Fewer programs, reused

Menu to call subprograms from TDF01

Debugger Assessor: ACEC

Report template for recording:
- Test results, execution time
- Comments on usability
- Commands used

More detailed scenarios

More comparable between systems

Program(s) for each scenario

Tasking, non-tasking separate

Debugger Assessor: AES Areas with Little Overlap

	AES	ACEC
Documentation	25	0
Source display	15	2
Error handling	13	0
Macros	9	2
Tracing	17	1
Private types	7	0
Heap	6	0
Debugging file IO	3	0
Performance	16	4
Capacity	13	1
Overall summary	10	0

Duplicate Assessors

Merge selected tests

Review AES approach

Review for easier portability

Assessors: AES only

C*	command language interpreter
E*	editor tests
N*	name expander tests
P*	pretty printer tests
R*	requirements analyses tests
T*	test support toolset tests
	test bed generator
	stub generator
	test coverage analysis
	timing analysis (profiler)
	assertion checker
V*	version and configuration control tests
X*	cross-reference tests

Assessors: Candidates for inclusion

- Profiler
- Test coverage analysis
- Cross reference
- Pretty printing
- Syntax based editing
- Test bed generator
- Name expander
- Stub generator
- Assertion checker

Assessors: Non-candidates

Command language interpreter

Editor tests (general)

Requirements analyses tests

Version and configuration control tests

Move to Reference System

Summary

- Interface
  - Database
  - Command file generator
- AES Performance Tests
  - Review and merge
- Analysis
- AES Assessors
  - Review and merge
  - Make others available